Subthreshold and Near-Threshold Techniques for Ultra-Low Power CMOS Design

by

James Anthony Kitchener
B.E.(Honours)(Computer Systems), B.MA.&Comp.Sc.,
The University of Adelaide

Thesis submitted for the degree of

Doctor of Philosophy

School of Electrical and Electronic Engineering,
Faculty of Engineering, Computer
and Mathematical Sciences
The University of Adelaide, Australia

July 2015
# Contents

## List of Figures

## List of Tables

## Abstract

## Statement of Originality

## Acknowledgements

## Chapter 1. Introduction

1.1 Sources of power consumption .............................................. 1

1.2 Applications ................................................................. 2

1.2.1 Smart Dust ................................................................. 2

1.2.2 Biomedical Implants .................................................. 3

1.2.3 Connected Standby ..................................................... 3

1.3 Thesis Structure .............................................................. 3

1.4 Publications ................................................................. 5

## Chapter 2. Characterising Subthreshold Gates

2.1 Characterising Devices ..................................................... 7

2.1.1 Non-traditional Devices ............................................... 8

2.2 Characterising Gates ..................................................... 9

2.2.1 Methodology .............................................................. 9

2.2.2 Baseline Results ....................................................... 11

2.2.3 Supply Voltage .......................................................... 11
Chapter 3. Characterising Subthreshold Adders

3.1 Metrics ................................................. 30
  3.1.1 Power-Delay Product and Energy per Operation .... 30
  3.1.2 Energy-Delay Product ................................ 31
3.2 Short Carry Chains ...................................... 31
  3.2.1 Methodology ....................................... 33
  3.2.2 Measurements ...................................... 34
  3.2.3 EPO and EPD ...................................... 37
  3.2.4 Reliability ......................................... 39
3.3 Adders .................................................. 43
  3.3.1 Ripple Carry Adder .................................. 43
  3.3.2 Tree Adders ........................................ 45
  3.3.3 Methodology ....................................... 45
  3.3.4 Measurements ...................................... 46
3.4 Further Work ........................................... 50
3.5 Conclusion ............................................. 51
3.6 Acknowledgement ....................................... 52

Chapter 4. Architectures for Subthreshold Operation .................................. 53

4.1 Motivation ............................................. 54
4.2 Asynchronous Design ........................................... 54
   4.2.1 Asynchronous at Subthreshold .......................... 57

4.3 Mixed Timing Domain Systems ................................. 58
   4.3.1 Globally Asynchronous Locally Synchronous Systems .... 59
   4.3.2 GALS at Subthreshold .................................. 61
   4.3.3 Mixed Timing Domain Topologies ...................... 61

4.4 Mixed Timing Domain Design ................................ 63
   4.4.1 Methods to Prevent Timing Failures ............... 64
   4.4.2 Conversion between Synchronous and Asynchronous Domains 68
   4.4.3 Procedure for Designing MTD Systems ............ 68
   4.4.4 Limitations ........................................... 79
   4.4.5 Application to Published Designs .................. 80

4.5 Further Work .................................................. 85

4.6 Conclusion ....................................................... 87

Chapter 5. Arbitrary Mixed Timing Domain Systems .......... 89

5.1 FIFOs in Literature ........................................... 90
   5.1.1 Comparison to New FIFOs ............................ 91

5.2 Mixed Clock FIFO .............................................. 92
   5.2.1 FIFO Operation ....................................... 93
   5.2.2 Data Validity Controller ............................... 97
   5.2.3 Full/Empty Detectors ................................ 99
   5.2.4 Timing Domain Analysis ............................... 101
   5.2.5 Resistance to Timing Failures ....................... 102

5.3 Asynchronous to Asynchronous FIFO ........................ 102
   5.3.1 Design ................................................ 103
   5.3.2 Operation ............................................ 104
   5.3.3 Resistance to Timing Failures ....................... 105

5.4 Hybrid Domains ............................................... 106
Chapter 6. A Wakeup Timer Based on Starved Transistors

6.1 Slow, Low-Power Clocks
   6.1.1 Slow, Low-Power Clocks in Literature

6.2 Target Application
   6.2.1 Electronic Smart Label Project
   6.2.2 Requirements
   6.2.3 Other Applications

6.3 Starved Gates
   6.3.1 How Starved Gates Work
   6.3.2 Characterising Starved Gates
   6.3.3 Environmental Analysis
   6.3.4 Tie Cells

6.4 The Buffered Starved Oscillator
   6.4.1 Simulation Methodology
   6.4.2 Simulation Results

6.5 Buffered Load
   6.5.1 Motivation
Chapter 7. Conclusions

Bibliography
# List of Figures

2.1 Gate testbed (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE) 9
2.2 Supply voltage vs propagation delay ................................. 12
2.3 Left: Supply voltage vs dynamic power. Right: energy consumed over a single transition .................................................. 12
2.4 Leakage of the two input gate $a \cdot b$ with respect to supply voltage . . . 13
2.5 Effect of supply voltage on worst case noise margin .................... 14
2.6 Temperature vs propagation delay for a NAND2 gate normalised to $30^\circ$ ................................................................. 15
2.7 Power consumption for a NAND2 gate normalised to $30^\circ$. Left: Dynamic power. Right: Leakage power ....................... 16
2.8 Noise margins for the INV and INV25 gates (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE) ....................................... 18
2.9 Voltage sweep for WP and WS corners ................................. 20

3.1 4-bit versions of the carry chains tested: (a) grey cells; (b) inverting grey cells; (c) valency 3 grey cells; (d) valency 3 inverting grey cells; (e) valency 4 inverting grey cells (6-bit). ............................................ 32
3.2 Cells used in the carry chains of prefix adders and their equivalent implementations .......................................................... 33
3.3 Testbed used for carry chain simulations .................................. 34
3.4 Carry chain average propagation delay from $C_{in}$ to $C_{out}$. Left: $V_{DD} = 0.3$ V Right: $V_{DD} = 1.8$ V ........................................... 35
3.5 Carry chain leakage power. Left: $V_{DD} = 0.3$ V Right: $V_{DD} = 1.8$ V 35
3.6 Carry chain average energy per switch from $C_{in}$ to $C_{out}$. Left: $V_{DD} = 0.3$ V Right: $V_{DD} = 1.8$ V ................................. 36
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>Leakage power for 6 bit carry chains</td>
<td>36</td>
</tr>
<tr>
<td>3.8</td>
<td>Energy per operation. Left: $V_{DD} = 0.3 , \text{V}$ Right: $V_{DD} = 1.8 , \text{V}$</td>
<td>38</td>
</tr>
<tr>
<td>3.9</td>
<td>Supply voltage vs energy per operation for 6 bit carry chains</td>
<td>38</td>
</tr>
<tr>
<td>3.10</td>
<td>Energy-delay product for 6 bit carry chains</td>
<td>39</td>
</tr>
<tr>
<td>3.11</td>
<td>Minimum supply voltage required for inverting grey cell carry chain</td>
<td>42</td>
</tr>
<tr>
<td>3.12</td>
<td>8-bit versions of the adders tested: (a) ripple-carry; (b) inverting ripple-carry; (c) PG ripple; (d) inverting PG ripple; (e) Sklansky; (f) inverting Sklansky; (g) valency 3 Sklansky.</td>
<td>44</td>
</tr>
<tr>
<td>3.14</td>
<td>Adder average propagation delay from $C_{in}$ to $C_{out}$ at $V_{DD} = 0.3 , \text{V}$ (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)</td>
<td>47</td>
</tr>
<tr>
<td>3.15</td>
<td>Adder average switching energy at $V_{DD} = 0.3 , \text{V}$ (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)</td>
<td>47</td>
</tr>
<tr>
<td>3.16</td>
<td>Adder leakage power at $V_{DD} = 0.3 , \text{V}$ (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)</td>
<td>48</td>
</tr>
<tr>
<td>3.17</td>
<td>Energy per operation at $V_{DD} = 0.3 , \text{V}$</td>
<td>48</td>
</tr>
<tr>
<td>3.18</td>
<td>Percentage of power attributable to leakage for 32 bit adders</td>
<td>49</td>
</tr>
<tr>
<td>4.1</td>
<td>A Mixed Timing Domain system consisting of two different clock domains</td>
<td>60</td>
</tr>
<tr>
<td>4.2</td>
<td>A GALS system with multiple synchronous domains communicating over an asynchronous channel</td>
<td>60</td>
</tr>
<tr>
<td>4.3</td>
<td>Point to point ring topology</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>Timing constraints of a D-flop</td>
<td>63</td>
</tr>
<tr>
<td>4.5</td>
<td>Implementation of clock guarding</td>
<td>66</td>
</tr>
<tr>
<td>4.6</td>
<td>Example system to illustrate the Mixed Timing Domain design procedure</td>
<td>68</td>
</tr>
<tr>
<td>4.7</td>
<td>Procedure step 1</td>
<td>70</td>
</tr>
<tr>
<td>4.8</td>
<td>Procedure step 2</td>
<td>71</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>Procedure steps 3 and 4</td>
<td></td>
</tr>
<tr>
<td>4.10</td>
<td>Procedure step 5</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>Procedure step 6</td>
<td></td>
</tr>
<tr>
<td>4.12</td>
<td>Procedure step 7</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>Procedure step 8</td>
<td></td>
</tr>
<tr>
<td>4.14</td>
<td>Procedure step 9</td>
<td></td>
</tr>
<tr>
<td>4.15</td>
<td>Mixed-clock FIFO stage from the Chelcea design. Diagram adapted from Fig. 6, [17] © 2004 IEEE</td>
<td></td>
</tr>
<tr>
<td>4.16</td>
<td>Flaw in Ono Design. Diagram adapted from [72] Figures 10 and 11, © 2009 IEEE</td>
<td></td>
</tr>
<tr>
<td>4.17</td>
<td>Flaw in Rahimian Design. Diagram adapted from Fig. 2 and Fig. 4 in [79], © 2010 IEEE</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of a FIFO</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>MC FIFO high level design</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>MC stage architecture</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Timing diagram for sending a word to the MC FIFO</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Timing diagram for retrieving a word from the MC FIFO</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>(a) Original Empty Detector (b) Nearly Empty Detector</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Data Validity Controller for MC FIFO</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>STG describing part of the non-synchronous portion of MC DVC</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>MC FIFO high level design</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>Top level architecture for a length 3 AA FIFO</td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>AA cell architecture</td>
<td></td>
</tr>
<tr>
<td>5.12</td>
<td>State transition graph for AA Data Validity Controller, reproduced from [17]. © 2004 IEEE</td>
<td></td>
</tr>
<tr>
<td>5.13</td>
<td>Timing Diagram for write and read operations on the AA FIFO</td>
<td></td>
</tr>
<tr>
<td>5.14</td>
<td>An SA FIFO stage</td>
<td></td>
</tr>
<tr>
<td>5.15</td>
<td>Data Validity Controller for AS FIFO</td>
<td></td>
</tr>
</tbody>
</table>
## List of Figures

5.16 STG for the asynchronous portion of the AS FIFO ........................................ 107
5.17 STG for the asynchronous portion of the SA FIFO ........................................ 108
5.18 Micrograph of fabricated chip implementing an earlier iteration of the FIFOs .......................................................... 109
5.19 Left: Latency for a combined put/get operation. Right: Normalised latency .......................................................... 110
5.20 Latency for the various FIFOs including synchroniser delay, normalised to the MC FIFO in the right graph .................................................. 111
5.21 Left: Throughput for a combined put/get operation. Right: Normalised throughput .......................................................... 112
5.22 Supply vs maximum sustained power consumption ......................................... 114
5.23 Percentage of energy attributed to leakage over a combined put/get operation .......................................................... 114
5.24 Leakage power consumption .......................................................... 115
5.25 Best and worst case power consumption for clock gating ................................... 116
5.26 Supply vs energy per operation .......................................................... 117

6.1 A conceptual diagram showing the physical construction of the smart label (from [77]) .......................................................... 127
6.2 A flow diagram showing the temperature logging behaviour of the smart label (from [77]) .......................................................... 127
6.3 (a) A fully starved inverter. (b) A semi-starved inverter ......................................... 129
6.4 A simplified starved inverter .......................................................... 131
6.5 Effect of supply voltage on the behaviour of the starved inverters. Left: Propagation delay. Right: Energy per transition ......................................... 137
6.6 Effect of temperature on the behaviour of the starved inverters. Left: Propagation delay. Right: Power consumption ......................................... 137
6.7 Maximum and minimum achievable output voltage levels ................................... 138
6.8 An oscillator built using starved gates .......................................................... 141
6.9 Waveform of a semi-starved oscillator .......................................................... 143
6.10 Waveform of a fully starved oscillator ........................................ 143
6.11 Effect of supply voltage on the behaviour of starved oscillators. Left: Frequency. Right: Combined power consumption ..................... 145
6.12 Effect of power consumption on the components of starved oscillators. 146
6.13 Effect of temperature on the behaviour of starved oscillators. Left: Frequency. Right: Combined power consumption ..................... 147
6.14 Effect of temperature on the frequency of oscillation of the semi-starved inverter with $V_{DD}$ varied between 1 and 1.5 V. ................. 148
6.15 Power consumption of an ordinary inverter with a DC voltage sweep applied to the input. .................................................. 153
6.16 Power consumption of individual transistors after application of voltage to gate ......................................................... 153
6.17 Testbench for load gate voltage sweep ........................................ 156
6.18 Power consumption of various load gates with a DC voltage sweep applied to the input at $V_{DD} = 1.2$ V ................................. 157
6.19 Power consumption of inverters with various lengths with respect to gate voltage .......................................................... 157
6.20 Power consumption of a semi-starved oscillator with various buffered loads ............................................................................. 158
6.21 Power consumption of various load gates with a DC voltage sweep applied to the input at $V_{DD} = 1.5$ V ................................. 159
6.22 Effect on starved transistor sizing on frequency ......................... 161
6.23 Effect of starved transistor width on power consumption. Left: $V_{DD} = 1.0$ V. Right: $V_{DD} = 1.5$ V ................................. 162
6.24 Frequency and power consumption of semi-starved oscillators tested with and without balancing pMOS transistor. .................... 163
6.26 Frequency of unbalanced and rail balanced oscillators .................. 165
6.27 Minimum voltage achievable by a semi-starved oscillator ................ 166
6.28 A starved oscillator with additional resistive load .......................... 167
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.29</td>
<td>Performance of a semi-starved oscillator with resistive load. Left:</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>Frequency. Right: power consumption.</td>
<td></td>
</tr>
<tr>
<td>6.30</td>
<td>Configuration of fabricated oscillators (from Kitchener and Phillips [51])</td>
<td>169</td>
</tr>
<tr>
<td>6.31</td>
<td>Micrograph of fabricated oscillators (from Kitchener and Phillips [51])</td>
<td>170</td>
</tr>
<tr>
<td>6.32</td>
<td>Supply voltage vs frequency of fabricated oscillators (from Kitchener and</td>
<td>172</td>
</tr>
<tr>
<td></td>
<td>Phillips [51])</td>
<td></td>
</tr>
<tr>
<td>6.33</td>
<td>Power consumption vs frequency of fabricated oscillators (from Kitchener</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>and Phillips [51])</td>
<td></td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Performance of a two input NAND gate</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Performance of inverter configurations at 0.3V, 0.5V and 1.8V</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Power consumption measurements of inverter configurations at 0.3V, 0.5V and 1.8V</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Unscaled results for an inverter</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Logic family behaviour at $V_{DD} = 0.3$ V (and $V_{DD} = 1.8$ V). All transistors are minimum width (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Comparative merits of gates at different supply voltages</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Minimum supply voltage for successful operation</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Parameters for Monte Carlo analysis</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Minimum supply voltage statistics after Monte Carlo simulation</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary of the merits of chains of gates compared to low-valency non-inverting logic</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Data representation in dual rail designs</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Summary of high-level design methodologies when applied to sub-threshold designs</td>
<td>86</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary of asynchronous design methodologies as applied to sub-threshold design</td>
<td>88</td>
</tr>
<tr>
<td>5.1</td>
<td>Output states for MC Data Validity Controller</td>
<td>98</td>
</tr>
<tr>
<td>6.1</td>
<td>Steady-state drain currents of the starved transistors in a fully-starved gate</td>
<td>134</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance characteristics of starved inverters at $V_{DD} = 1.2$ V</td>
<td>136</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance characteristics of a semi-starved inverter at various corners</td>
<td>139</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance characteristics of a fully starved inverter at various corners</td>
<td>140</td>
</tr>
</tbody>
</table>
List of Tables

6.5 Performance characteristics of starved oscillators at $V_{DD} = 1.2$ V . 144
6.6 Performance characteristics of a semi-starved oscillator at various corners ................................................. 149
6.7 Performance characteristics of a fully starved oscillator at various corners ......................................................... 150
Abstract

The miniaturisation of electronic circuits allows the potential for new applications, such as smart-dust or the Internet of Things. However, the design of batteries has not improved at the same rate as CMOS technology, so circuits need to be designed for improved energy efficiency to enable new form factors and applications.

To address these issues, the use of subthreshold and near-threshold supply voltages is proposed. Throughout this thesis, the nature of what makes a design suitable for subthreshold use is examined. This work starts at the gate level, where the effects of transistor geometry and valency are examined. The levels of abstraction are progressively increased until high level architectures are considered, where quasi-delay-insensitive and globally-asynchronous locally-synchronous designs are argued as suitable for designing reliable systems. To assist in this, a methodology for partitioning systems into separate timing domains is proposed, and applied to published designs.

The underlying theme throughout the exploration of subthreshold technology is the effects and mitigation of process and environmental variation, to which designs are increasingly susceptible as the supply voltage is lowered. This vulnerability affects all levels of design, from the widths of individual transistor to the choice of overall architectures, where a fundamental issue is the ability to determine when a unit of work has been performed.

Not all applications respond well to the scaling of supply voltage. To address this, an alternative approach is considered where the system spends much of its lifetime in a powered-down state, being woken at appropriate intervals by a wakeup timer. As power consumption is a function of frequency, this timer seeks to achieve energy efficiency by maximising the period of oscillation. Despite the higher supply voltages considered, the themes of environmental and process variation continue, as the wakeup timers examined share similarities to subthreshold designs. Two of the proposed timers have been fabricated and are compared to simulated results and other published work.
Statement of Originality

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works. I also give permission for the digital version of my thesis to be made available on the web, via the University’s digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

______________________________  ____________________
Signed  Date
Acknowledgements

I would like to thank my supervisors, Dr. Braden Phillips and A/Prof. Mike Liebelt, for the advice and mentoring they have provided over my many years of candidature.

I acknowledge the help of the School of Electrical and Electronic Engineering, for funding my Divisional Scholarship and for the provision and support of the equipment and software needed for my research.

I owe a debt of gratitude to Prof. David Blaauw of the University of Michigan, for funding the initial work on subthreshold logic and for starting me on the eventual direction that my research followed.

The assistance that Dr. Braden Phillips and Robert Moric provided in preparing my chip for fabrication was invaluable and allowed me to submit the chip within a tight deadline.

Finally I would like to thank my parents, for the support they have given me. I would not have been able to complete this long and difficult journey without their encouragement.

James
Chapter 1

Introduction

Advances in electronic technology have resulted in computers becoming radically smaller. Millimetre scale computing, where the target size of the device is only $1 \text{ mm}^3$ is the next step in this trend [109]. As this volume includes components responsible for storage or generation of energy, it imposes severe constraints on the power consumption of the device. Compounding this problem, the development rate of electronic circuitry has consistently exceeded that of batteries [58]. For long lasting electronics in millimetre scale applications, improvements in service life will have to come from a more energy efficient chip, rather than an improved battery.

1.1 Sources of power consumption

The power dissipation of a CMOS circuit can be modelled as [110, page 188]

$$ P_{\text{total}} = P_{\text{static}} + P_{\text{dynamic}} $$

(1.1)

Where these terms are defined as:

$$ P_{\text{static}} = I_{\text{static}} V_{DD} $$

(1.2)

$$ P_{\text{dynamic}} = \alpha C V_{DD}^2 f $$

(1.3)

$\alpha$ is the activity ratio, or the proportion of transistors that transition in a single period; $C$ is the capacitance of the circuit, $V_{DD}$ is the supply voltage and $f$ is the frequency of operation. For a synchronous circuit this would be the clock frequency, but it is harder to define for an asynchronous circuit. The static current, $I_{\text{static}}$ is
the supply current observed whenever the circuit is at steady state and will also be referred to as leakage. Low power operation is obtained by reducing any or all of these factors while still meeting the desired performance characteristics.

A dramatic reduction in power consumption can be achieved by reducing $V_{DD}$, as this reduces both static and dynamic power. As a consequence of dropping the supply voltage, the maximum frequency of the circuit is reduced as subthreshold and near-threshold systems cannot work as quickly. The impact of supply voltage on frequency and power consumption is examined in Chapter 2.

**Definition**

A circuit is said to have a *subthreshold* supply if the voltage source powering the circuit is less than the threshold voltage of the transistors ($V_{DD} < V_{th}$). Likewise a *near-threshold supply* is one where the supply voltage is at or slightly greater (within a few hundred mV) of the threshold voltage. The remaining region is called the *superthreshold* region and this is where most conventional circuits operate.

### 1.2 Applications

#### 1.2.1 Smart Dust

Smart dust [109] is a promising application for millimetre-scale devices. Envisioned as remote sensors, communicating over ad-hoc mesh networks, smart dust could be useful for environmental or infrastructure monitoring. As it is designed to be cheap, plentiful and expendable, reliability of individual motes of smart dust is unimportant, providing aggregate reliability is maintained. Millimetre-scale technology could also be embedded in textiles to create smart clothing [74]. The Internet of Things [53] is a similar approach which aims to connect arbitrary objects to the internet. The ability to reduce power consumption through the use of subthreshold supply voltages may prove beneficial for battery powered devices that are intermittently connected.
1.2.2 Biomedical Implants

For devices embedded deep within the body, changing an included battery would be impractical and surgically invasive. Rather than a battery, energy harvesting could be used to generate the required power [71]. The amount of power that can be generated is dependent on the type and size of the power generation device [78] but is typically less than 1 mW/cm². For small scale devices only a small amount of power can be generated so it would need to be used judiciously [18].

1.2.3 Connected Standby

There are also opportunities for improving the energy efficiency of larger devices. Computer notebooks and mobile phones are expected to run increasingly heavy work loads, while maintaining an acceptable battery life. Even when the screen is off and the device is not being used, there is still a need to perform computations. Operating at a near-threshold supply voltage would allow non-time critical operations to be performed while minimising the battery drain. When the user interacts with the device again, the supply voltage can be restored to its normal level.

Intel uses a form of connected standby, called Smart Connect [33], in its processors, although it is implemented differently. Intel chooses a normal supply voltage to complete the tasks quickly before returning to a powered down state.

Portable devices are not the only beneficiary of reduced power consumption. The One Watt Initiative [69] seeks to reduce the standby power of appliances to below one watt. Despite the low power consumption, devices in standby are still expected to be responsive. For example, a television must keep the remote control sensor active at all times while a desktop computer may keep its network controller active to respond to Wake-On-LAN events. Maintenance tasks can run slowly overnight to ensure the system is responsive when actively used.

1.3 Thesis Structure

The first part of this thesis, Chapters 2 and 3, examines the consequences of operating circuitry with subthreshold or near-threshold supply voltages. Chapter 2
begins the analysis at the logic gate level. The reliability of subthreshold gates is examined in terms of the variation to which they are exposed, with attention drawn to both process and environmental variation. A study of larger modules, composed of multiple subthreshold gates is performed in Chapter 3. Using adders as a case study, the trade-offs between latency and energy efficiency are examined. Here process variation is studied from a different perspective, measuring the reliability of a design in terms of the minimum supply voltage required to operate.

The theme of reliability and process variation in subthreshold circuits is continued in the second part, Chapters 4 and 5. The most serious issue in designing reliable systems is that of intra-die process variation, by which the underlying transistors in the design operate at different speeds, violating timing assumptions. Chapter 4 proposes a solution to the problem by partitioning the design into smaller timing domains, applying the principles of Globally Asynchronous Locally Synchronous (GALS) designs to subthreshold designs. A methodology to assist in the partitioning of a design into timing domains is proposed. Chapter 5 describes a family of FIFOs which can be used to transmit information safely between timing domains. These FIFOs are evaluated at subthreshold and near-threshold supply voltages and are used as case studies to evaluate synchronous and asynchronous design techniques at subthreshold.

The final part, Chapter 6, addresses the problem of energy efficiency from a different perspective. The power consumption of a system can be reduced by powering down components when they are not needed, however a small subsystem must always remain active in order to wake the design when work needs to be performed. To address this, Chapter 6 proposes a wakeup timer which reduces power consumption by minimising the frequency at which the clock oscillates. This is achieved by inserting transistors whose gate voltages are fixed such that they always operate in the subthreshold region, a process known as starving. Despite the differences in supply voltage and transistor geometry between starved gates and ordinary subthreshold logic, substantial similarities are observed in the way the two techniques respond to process and environmental variation.
1.4 Publications


Chapter 2

Characterising Subthreshold Gates

One way to begin an investigation into subthreshold logic is to determine how it behaves at a gate level. A coarse summary would be that a gate’s behaviour at subthreshold is similar to superthreshold, except that it is markedly slower and consumes significantly less power. While a useful first order approximation, this glosses over the subtleties and minor differences between the two operating regimes and leads to traps where behaviour isn’t as expected.

This chapter splits the low level characterisation of subthreshold logic into three categories: devices, gates and logic families. In each case a survey of published works is performed. A subthreshold gate is examined in detail in Section 2.2 and a logic family is characterised in Section 2.3.1. The analysis of larger modules is performed in Chapter 3 with a case study on adders.

2.1 Characterising Devices

Models for subthreshold behaviour of CMOS transistors are proposed by [100] and [103]. [46] applies the principles of logical effort to find optimum sizing in the subthreshold regime. A theoretical approach to minimum energy operation is employed by [12].

The design of transistors has also been evaluated at subthreshold. [76] showed that the use of conventional devices isn’t an optimal solution and proposed optimised devices for subthreshold operation. [36] investigated device scaling at subthreshold. Both papers conclude that reduced doping improves performance, although [36] also proposes the use of transistors with longer than minimum length channel regions.
2.1 Characterising Devices

In this thesis, the transistors used will consist of ordinary silicon-based MOSFETs designed for superthreshold applications. The goal was to explore low cost applications of subthreshold design. Therefore high volume, conventional CMOS processes were selected. The requirement of exotic devices, or non-standard doping levels would create additional barriers to adoption.

2.1.1 Non-traditional Devices

An alternative approach is to examine non-conventional devices to see if they are better suited to subthreshold or near-threshold operation.

Both single [24, 41] and double gated [81] FinFETs have been studied. The common conclusion of these papers is that FinFETs operating at subthreshold supply voltages had reduced switching delay, improved energy efficiency and greater robustness to process variation than MOSFET based designs. As many foundries are transitioning towards FinFETs in the development of newer process technologies, the ability to operate FinFETs below the threshold voltage is important to ensure long term viability for subthreshold design.

[82] investigated a carbon nanotube transistor variant for use as a footer for power gating. The design achieved significantly lower leakage, and hence battery life, than a design constructed using a conventional silicon transistor. Dynamic threshold logic, which adjusts the threshold voltage through the use of substrate biasing was tested by [95] and was found to achieve markedly faster switching speeds at a similar energy per transition compared to conventional designs. [48] investigated double gate transistors and claimed that they were well suited to operation at subthreshold supply voltages. Although various configurations were tested, no comparison against a baseline implementation was made. Floating gate transistors are used in the context of a D-latch in [3], but are poorly suited to subthreshold operation. When compared to static CMOS gates operated at subthreshold supply voltages, the switching speed of the floating gate design is up to 18 times faster, but this comes at the cost of a power consumption increase of up to 1000 times. If floating gate devices are compatible when incorporated into an otherwise conventional design, then sparing use may be beneficial on the critical timing path of the design to meet timing requirements. In order to maintain a suitably low power consumption,
Chapter 2 Characterising Subthreshold Gates

Figure 2.1: Gate testbed (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)

The primary motivation for using subthreshold supply levels, the use of floating gate devices must be minimised.

2.2 Characterising Gates

The next higher level of abstraction above the device level is the gate level. Gates are comprised of multiple devices combined to perform a logic operation. The following sections report the results of a detailed study into gates at subthreshold and near-threshold supply levels, in order to understand their behaviour. Relevant supporting results have been incorporated from the literature in which aspects of gate behaviour have hitherto been studied in isolation but never before collated into a single document.

2.2.1 Methodology

The methodology which I used previously in [10], which itself was based on [110, p. 5], was adopted for the remainder of this chapter. The test bed used is shown in Figure 2.1. The same type of gate was used for the device under test, source and load gates. The inputs to each of these gates were arranged so that they operated as inverters. The power of the device under test was isolated using a separate power rail. The circuits were simulated in Synopsys HSPICE D-2010.03-SP2.

A temperature of 30° Celcius was adopted. This is in contrast to the 70° value used in [10]. The power consumption of subthreshold and near-threshold circuits is so low that self-heating effects are negligible and the ambient temperature is the only influence. Given the results in Section 2.2.4, this is actually a more conservative value...
2.2 Characterising Gates

at subthreshold supply voltages. Also, as one of the target applications described in Chapter 1 was medical implants, the choice of a temperature closer to that of core body temperature of approximately 37°C for many animals allows for more relevant results.

Two-input NAND gates were examined to explore how subthreshold and near-threshold logic differs from superthreshold. The NAND gate was minimum sized with all transistors sharing the same width of 0.36 µm and length 0.18 µm. The circuits were simulated using a TSMC 180 nm process, with model libraries derived from a MOSIS test lot*. 

A fanout value of 4, reflected by \( H = 4 \) in Figure 2.1, was used for time measurements and a fanout of 1 was used for power measurements. The testbed simulates fanout by successively scaling the gates. As a consequence, the capacitance of the gate is artificially enlarged. Given dynamic power consumption is proportional to capacitance†, the power measurements at a fanout value of 4 would be artificially high.

The variability of leakage power, shown in Section 2.2.3, poses a challenge for the measurement of dynamic power. Simply subtracting the average or worst case leakage from total power consumption can easily result in the value for dynamic power being negative. For individual gates or homogeneous structures, leakage must be determined based upon the state of the system. By measuring the system without a change in inputs, leakage power can be isolated and is recorded separately for each possible state.

Total energy consumption was measured over an interval that extends from the start of the input transition until after circuit returns to steady state. As dynamic energy continues to be dissipated after the conclusion of the output transition, this ensures that power consumption is not underestimated. The leakage energy for the final state is then subtracted from total energy to provide the dynamic energy of the gate under consideration. Peak power consumption is then obtained by dividing by the interval between the input transition and the corresponding output transition. Leakage is measured on a per-state basis to account for its variability.

†See Equation (1.3) on page 1
2.2.2 Baseline Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>Sub-Vt</th>
<th>Near-Vt</th>
<th>Super-Vt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage (V)</td>
<td>0.3</td>
<td>0.5</td>
<td>1.8</td>
</tr>
<tr>
<td>Average Propagation Delay (ns)</td>
<td>649.1</td>
<td>12.32</td>
<td>0.1356</td>
</tr>
<tr>
<td>Rise Time (ns)</td>
<td>787.1</td>
<td>13.89</td>
<td>0.2846</td>
</tr>
<tr>
<td>Fall Time (ns)</td>
<td>541.8</td>
<td>9.459</td>
<td>0.1834</td>
</tr>
<tr>
<td>Peak Dynamic Power (nW)</td>
<td>0.5188</td>
<td>80.59</td>
<td>64,180</td>
</tr>
<tr>
<td>Static Power (pW)</td>
<td>1.888</td>
<td>3.928</td>
<td>37.94</td>
</tr>
<tr>
<td>Energy per Operation (fJ)</td>
<td>0.2807</td>
<td>0.7800</td>
<td>10.19</td>
</tr>
<tr>
<td>Low Noise Margin (mV)</td>
<td>96.36</td>
<td>190.3</td>
<td>451.3</td>
</tr>
<tr>
<td>High Noise Margin (mV)</td>
<td>134.4</td>
<td>233.6</td>
<td>831.2</td>
</tr>
</tbody>
</table>

Table 2.1: Performance of a two input NAND gate

To allow trends to be observed on the same scale, graphs have been normalised to a common case unless otherwise specified. The voltage range was split into three broad regions, depending on the supply voltage relative to the threshold voltage. Unscaled performance metrics are shown in Table 2.1.

2.2.3 Supply Voltage

The effect of supply voltage on propagation delay in a NAND2 gate is shown in Figure 2.2 on the next page. No normalisation has been performed in this subsection. In the subthreshold region propagation delay is exponential with respect to supply voltage with the increase slowing in the near-threshold region and even further in the super-threshold region. Plots of rise and fall time display the same trend.

The left graph of Figure 2.3 shows the average power consumption over the transition interval. This is equivalent to the power consumption incurred by clocking the device at the maximum speed possible. The effect of supply voltage on dynamic power consumption is much larger than the quadratic relationship suggested by
2.2 Characterising Gates

Figure 2.2: Supply voltage vs propagation delay

Figure 2.3: Left: Supply voltage vs dynamic power. Right: energy consumed over a single transition
Equation (1.3) on page 1 due to the simultaneous increase in clock frequency. The right hand graph of Figure 2.3 normalises the power consumption to a unit transition and depicts a slower, quadratic increase.

Leakage power, shown in Figure 2.4 is also quadratically affected by variations in supply voltage although the slope is shallower than for dynamic energy. This is as predicted by Equation (1.2) on page 1 after applying Ohm’s law to \( I_{\text{static}} \). That Ohm’s law is applicable suggests that the transistors are operating in the linear region, which can be modelled as a resistor [110, page 70]. The differing behaviour of leakage is caused by the use of minimum sized transistors and by the choice of 30° as the test temperature. Temperatures beyond room temperature exhibit greater variation and this is also apparent at very low temperatures, although the relative ordering of the logic states is different. The discrepancy depends on the gate topology, as I found in [10] that the difference between best and worst case was almost a factor of 10 for the NOR2 gate. This discrepancy between leakage states is explored further in Section 2.2.5.
2.2 Characterising Gates

At a supply voltage of 0.2V, leakage power accounts for approximately 4% of total power consumption, although the slow rise accounts for its impact quickly becoming negligible at supply voltages greater than 0.3V in this low leakage 180 nm process.

In the subthreshold and near-threshold regions, both the higher and lower noise margins increase linearly as shown in Figure 2.5. The low noise margin does not begin to level off until well into the super-threshold region. For an existing design, noise margin concerns can be easily rectified by increasing the supply voltage, providing the resulting increase in power consumption is tolerable. If this is not acceptable, alternatives are presented in Section 2.2.5.

### 2.2.4 Temperature

Compared with superthreshold, the temperature response at subthreshold is markedly different. Propagation delay, shown in Figure 2.6, and transition times are faster at high temperatures than they are at low temperatures. The behaviour of near-threshold circuits reflects that of superthreshold, although the temperature sensitivity is reduced.

![Figure 2.5: Effect of supply voltage on worst case noise margin](image-url)
If a synchronous subthreshold system is expected to operate over a wide temperature range, care must be taken with the clock. The clock frequency must either scale appropriately with temperature, or it must be specified with a large amount of slack.

The effects of temperature on power consumption is shown in Figure 2.7. At subthreshold, dynamic power consumption is exponentially dependent on temperature. Near-threshold and superthreshold supplies do not exhibit this behaviour and demonstrate much smaller variation.

Static power consumption behaves similarly at subthreshold as it does at superthreshold, as depicted in the right graph of Figure 2.7. In all cases temperature has a significant effect on leakage power.

The combined effects of an increase in dynamic and leakage power indicate that subthreshold battery life will be markedly shorter at high temperatures. Low noise margin is reduced by up to 15% relative to room temperature at high temperatures, suggesting decreased reliability. High noise margin is not significantly affected.
2.2 Characterising Gates

![Graph showing power consumption for a NAND2 gate normalised to 30\(^\circ\). Left: Dynamic power. Right: Leakage power.]

**Figure 2.7:** Power consumption for a NAND2 gate normalised to 30\(^\circ\). Left: Dynamic power. Right: Leakage power

### 2.2.5 Sizing

To study the effects of transistor sizing on gate behaviour, three inverter configurations were chosen. INV has minimum sized transistors with a width of 0.36 \(\mu\)m and a length of 0.18 \(\mu\)m. The pMOS transistor of the INV25 variant has a width of 0.9 \(\mu\)m, 2.5 times that of the nMOS transistor. The INV2xl configuration has transistors of minimum width and a length of 0.36 \(\mu\)m, twice that of the other gates.

The performance metrics of the various inverters are contained in Tables 2.2 and 2.3. The INV25 gate has superior propagation delay, rise and fall times compared to the INV variant, although this comes at the cost of high power consumption. An additional consequence of sizing the transistors equally is an asymmetric leakage power.

The lower energy per operation observed in the INV gate mirrors the result in [12], in which it was found that minimum device sizing was theoretically optimal for minimum energy operation. In contrast to this, [52] found that increasing the width of the pMOS transistor non-minimum sizing produced superior yields and allowed lower supply voltages to be used for a given yield.
### Chapter 2 Characterising Subthreshold Gates

<table>
<thead>
<tr>
<th>Vdd</th>
<th>INV</th>
<th>INV25</th>
<th>INV2x1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Propagation delay (ns)**

- **0.3 V**: 473.8, 433.6, 1145
- **0.5 V**: 9.043, 7.792, 22.75
- **1.8 V**: 0.1012, 0.1071, 0.3044

**Rise time (ns)**

- **0.3 V**: 663.1, 517.8, 1268
- **0.5 V**: 11.38, 8.259, 23.90
- **1.8 V**: 0.2528, 0.1742, 0.6750

**Fall time (ns)**

- **0.3 V**: 556.4, 418.2, 1147
- **0.5 V**: 7.117, 6.295, 16.66
- **1.8 V**: 0.1395, 0.1828, 0.3511

### Table 2.2: Performance of inverter configurations at 0.3V, 0.5V and 1.8V

<table>
<thead>
<tr>
<th>Vdd</th>
<th>INV</th>
<th>INV25</th>
<th>INV2x1</th>
<th>INV</th>
<th>INV25</th>
<th>INV2x1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dynamic power consumption (pW)**

- **0.3 V**: 0.4177, 0.4813, 0.01237
- **0.5 V**: 67.78, 162.8, 27.09
- **1.8 V**: 30,470, 2,322,000, 311,700

**Leakage Power High Output (pW)**

- **0.3 V**: 1.889, 1.889, 0.5697
- **0.5 V**: 3.928, 3.928, 1.203
- **1.8 V**: 37.94, 37.94, 10.48

**Energy per transition (fJ)**

- **0.3 V**: 0.1735, 0.2226, 0.0114
- **0.5 V**: 0.5060, 0.9009, 0.4810
- **1.8 V**: 3.401, 261.6, 92.24

**Leakage Power Low Output (pW)**

- **0.3 V**: 0.8453, 1.541, 0.516
- **0.5 V**: 3.114, 3.114, 1.092
- **1.8 V**: 16.06, 25.88, 9.311

### Table 2.3: Power consumption measurements of inverter configurations at 0.3V, 0.5V and 1.8V

---

Page 17
2.2 Characterising Gates

Noise margins for the INV and INV25 are shown in Figure 2.8. The gate with equally sized transistors has a superior low noise margin whereas INV25 has a better high noise margin. From a reliability perspective, INV25 can be considered preferable due to its smaller indeterminate region, in which an input voltage cannot produce a legal digital logic output level, supporting [52].

The effect of longer transistors in the INV2xl configuration produced more interesting results. As expected, longer transistors led to inferior propagation delay and transition times, however doubling the length of the transistors produced markedly different results at subthreshold compared to superthreshold. Dynamic power consumption and energy per transition were significantly reduced at subthreshold and markedly increased at superthreshold. In the near threshold region lower power and energy were also observed, albeit at a much reduced margin. This suggests that long transistors are an effective way of reducing both dynamic and static power consumption at subthreshold providing performance targets can be met. The benefits of longer length at subthreshold have also been noticed by [36], which used longer transistors at subthreshold to keep power consumption at an acceptable level.
Chapter 2 Characterising Subthreshold Gates

The effect of process scaling on the physical dimensions of gates needed to achieve minimum energy operation has been studied by Bol et al. [11]. Leakage power is increasingly a problem as smaller feature sizes are adopted [49], a trend which has been observed at both superthreshold and subthreshold supply voltages. To mitigate this, Bol adopted longer gates, achieving an energy reduction of 65% at a 22 nm node. As an additional benefit, the minimum supply voltage required for successful operation also decreased as length was increased, enhancing the reliability of the devices.

Although many of the trends established at superthreshold apply at subthreshold and near-threshold, this is not universal. For optimum performance gate configurations need to be retested at subthreshold. The lack of a configuration optimum at all supply voltages is problematic for circuits expected to operate at both superthreshold and sub or near-threshold. A choice prioritising one of the target operating regions must be made.

Keane et al.[45] investigates subthreshold sizing by modelling the width of stacked transistors needed to optimise current drivability. For a given stack depth, it was found that sizing the transistors equally was the most effective approach and that the ratio of the width stacked transistors compared that of a single transistor was greater at subthreshold than it is at a conventional superthreshold voltage. This increase in ratio is more pronounced at smaller process nodes, moving from 18% at 130 nm to 40% at 45 nm. The benefits of such an approach are a reduction of latency of up to 10% compared to sizing the stacks at a conventional superthreshold voltage, however there was no mention of the effect that the modified sizing would have on power consumption. The choice of a conventional superthreshold pMOS to nMOS width ratio of 1.5 highlights the focus on latency, as minimum sized devices achieve lower energy consumption [12].

2.2.6 Process Variability

To assess the impact of process variation an inverter was simulated in typical (TM), worst power (FF/WP) and worst speed (SS/WS) corners. The simulations were performed using an AMS 0.35 µm process, as corners for the TSMC 180 nm process were not available. As a result, different supply voltages were used to target
equivalent performance ranges. In [10], I addressed this lack of corners by defining variability solely as a variation in threshold voltage, but a broader definition of variability may provide different results. In Chapter 3, variability is simulated using the TSMC 180 nm process, but the simulation parameters were estimated based on extrapolations of the behaviour of the AMS 0.35 µm process corners.

The measurements for the various corners were all performed at a fixed temperature of 30°C. Allowing for temperature variations would produce even greater variability although care needs to be taken that temperatures appropriate to the voltage range are used as Section 2.2.4 showed that the way circuits react to temperature depends heavily on supply voltage.

The results are shown in Figure 2.9. Variability in propagation delay is significantly increased when operating at subthreshold voltages. This behaviour has been widely observed in literature. [35, 65, 89, 106, 115].

A number of approaches have been proposed to mitigate the increase in variability. [41] suggests that FinFET based designs offer reduced variability compared to conventional designs. Asynchronous circuit designs [38], in which logic is permitted to
work at its own pace with tolerance towards variation, is a commonly studied architectural approach to solve this problem. [23] proposes several asynchronous logic styles based on completion detection and current sensing. [13] uses multiple bundled delay paths to mitigate intra-die variation. Another solution to the problem of high variability, Globally Asynchronous Locally Synchronous (GALS), which splits a design into multiple independent timing domains, will be presented in Chapter 5.

### 2.2.7 Gate Design

Within a logic family there are multiple ways to design the more complicated gates. Not every design that works at superthreshold can scale down to low voltages, as [86] found that a gate relying on weak feedback could not operate in the subthreshold and near-threshold regions. This study compared a proposed C-element against the non-functional weak-feedback design, but no attempt was made to test against designs that may prove more appropriate, such as those compared in [92].

The comparative merits of flip-flop designs in the subthreshold regime have been studied in [4, 5, 16]. The recommended flip-flop depends on the emphasised metric used in the comparison. [5] favours performance and power consumption, while [16] prefers robustness. While proposing a latch for near-threshold use, [98] provides a detailed comparison of several other published designs.

SRAM design at subthreshold has been extensively studied, with many designs proposed [14, 50, 57, 105] but quantitative comparisons between the proposals are rarely performed.

Several full adder designs were investigated in [31]. The recommended full adder depends on design requirements and the target supply voltage.

### 2.3 Characterising Logic Families

While several logic families have been proposed for use at subthreshold, the characterisation of them is rarely published. [104] includes power measurements for a variety of gates, but does not include other metrics. In [10], I undertook a study of a complete logic family suitable for implementing arithmetic designs. This chapter provides an extended treatment of the first half of the latter paper.
2.3 Characterising Logic Families

The investigation of logic families at subthreshold is primarily done in one of two ways. The first is to use an individual logic gate as a case study. This approach is used by [15, 39]. The danger in this method is that the gate studied may not be representative of typical gates within the family. This is highlighted in [118], where although complementary pass-transistor logic had previously been found to produce more power efficient full-adder circuits, this result was not replicated when extended to other gates. Dual rail logic [97] provides an exaggerated example. By representing a logic state with a pair of wires, at most one of which can be high, it is possible to implement an inverter simply by swapping the two wires, with no transistors required. On the other hand, to perform a NAND2 operation requires 14 transistors when implemented using a static CMOS methodology, rather than 4 for a conventional CMOS design.

The more reliable approach to analysing logic families is to repeat the analysis for a variety of gates. This eliminates the risk of choosing a non-representative gate and allows more robust conclusions to be drawn. This is performed within this section and by [10] and [104] at subthreshold and [118] at superthreshold supply levels.

For more exotic logic families, such as Null Convention Logic [27] there may not be one-to-one correspondence between logic gates. In these situations, a third approach can be used. Rather than examine circuits at the gate level, the logic family is tested as part of a large design and compared against a baseline implementation. This approach is used by [43]. Care must be taken as the relative merits of the logic family may be exaggerated by asymmetric optimisation. The conclusions drawn by this approach may also be limited to the characteristics of the large design.

2.3.1 A Logic Family for Subthreshold

To explore the effects on conventional CMOS logic, a family of gates was simulated at both subthreshold and superthreshold supply voltages. The methodology from Section 2.2.1 was adopted, however the simulations were performed at 70°C, like those of [10]. This logic family had previously been analysed in [10], however minor corrections have been made. Although the previous study normalises all measurements to that of an inverter, the actual values for the inverter were accidentally omitted. Corrected figures for the grey cell at 0.3V are now provided.
Chapter 2  Characterising Subthreshold Gates

For the full adders, a 28 transistor static CMOS topology [110, page 640] was chosen. The inverting full adder is based on the same design, with the removal of the output inverters. The grey cell is a block common in the design of adders and will be discussed further in Chapter 3.

The parasitic delay has been given in units of $\tau$, where $\tau$ is one fifth of the FO4 inverter delay at $V_{DD} = 0.3$ V or 1.8 V as appropriate [110, page 164]. Logical effort is given in units of $\tau$ per fanout. These values were obtained from the average of rising and falling edges of the slowest input transition except for the full adder and grey cells for which the carry-in to carry-out transitions were observed.

The dynamic switching energy was measured as the average for the rising and falling transitions of the input with the worst-case switching energy. The leakage power was recorded for the input state with the highest leakage.

<table>
<thead>
<tr>
<th>Metric</th>
<th>$V_{DD} = 0.3$ V</th>
<th>$V_{DD} = 1.8$ V</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO4 Propagation delay (ns)</td>
<td>290.9</td>
<td>0.1157</td>
</tr>
<tr>
<td>Energy per Switch (fJ)</td>
<td>0.1406</td>
<td>5.343</td>
</tr>
<tr>
<td>Leakage Power (pW)</td>
<td>6.998</td>
<td>116.8</td>
</tr>
</tbody>
</table>

Table 2.4: Unscaled results for an inverter

The results are shown in Table 2.5 on the next page normalised to the behaviour of an inverter. To provide an indication of magnitude, the raw figures for the inverter are displayed in Table 2.4.

At the supply voltage tested, the superthreshold numbers for logical effort and parasitic delay in this table correspond well with nominal values often used for hand estimation [110, page 167]. When normalised against $\tau$ at 0.3 V, the subthreshold numbers differ in some interesting ways. The subthreshold parasitic delays are generally worse; however the logical effort for the inverter, NAND gates and full adder improve. The 3-input NAND has logical effort almost equal to the 2-input NAND. Hence for these gates, fanin and fanout have less influence on delay at subthreshold voltage but the no-load delay per stage is increased. This suggests
2.3 Characterising Logic Families

<table>
<thead>
<tr>
<th>Cell</th>
<th>Icon</th>
<th>Parasitic Delay $p [\tau]$</th>
<th>Logical Effort $g [\tau/\text{fanout}]$</th>
<th>Energy per Switch vs. inv</th>
<th>Leakage Power vs. inv</th>
</tr>
</thead>
<tbody>
<tr>
<td>inv</td>
<td>$\nabla$</td>
<td>1.134 (0.676)</td>
<td>0.961 (1.079)</td>
<td>1.000 (1.000)</td>
<td>1.000 (1.000)</td>
</tr>
<tr>
<td>nor2</td>
<td>$\nabla$</td>
<td>2.327 (2.188)</td>
<td>1.734 (1.662)</td>
<td>1.860 (1.853)</td>
<td>1.999 (2.000)</td>
</tr>
<tr>
<td>nand2</td>
<td>$\nabla$</td>
<td>1.526 (1.481)</td>
<td>1.337 (1.401)</td>
<td>1.785 (1.880)</td>
<td>0.999 (1.000)</td>
</tr>
<tr>
<td>xnor2</td>
<td>$\nabla$</td>
<td>6.145 (6.012)</td>
<td>3.817 (3.436)</td>
<td>4.746 (4.991)</td>
<td>3.957 (3.875)</td>
</tr>
<tr>
<td>xor2</td>
<td>$\nabla$</td>
<td>6.268 (6.008)</td>
<td>3.713 (3.468)</td>
<td>4.728 (4.924)</td>
<td>3.363 (3.230)</td>
</tr>
<tr>
<td>nand3</td>
<td>$\nabla$</td>
<td>2.825 (2.110)</td>
<td>1.322 (1.470)</td>
<td>2.122 (2.270)</td>
<td>1.217 (1.065)</td>
</tr>
<tr>
<td>nor3</td>
<td>$\nabla$</td>
<td>4.191 (3.842)</td>
<td>2.137 (2.013)</td>
<td>2.024 (2.209)</td>
<td>2.997 (3.000)</td>
</tr>
<tr>
<td>oai21</td>
<td>$\nabla$</td>
<td>3.409 (3.523)</td>
<td>1.913 (1.775)</td>
<td>2.101 (2.278)</td>
<td>1.856 (1.678)</td>
</tr>
<tr>
<td>aoi21</td>
<td>$\nabla$</td>
<td>3.311 (3.328)</td>
<td>1.906 (1.780)</td>
<td>2.182 (2.265)</td>
<td>1.998 (2.000)</td>
</tr>
<tr>
<td>fulladd ($C_{out}$)</td>
<td>$\nabla$</td>
<td>9.394 (9.365)</td>
<td>3.467 (3.732)</td>
<td>9.685 (10.636)</td>
<td>5.040 (4.101)</td>
</tr>
<tr>
<td>fulladdi ($C_{in}$)</td>
<td>$\nabla$</td>
<td>6.078 (6.591)</td>
<td>7.474 (7.867)</td>
<td>6.583 (7.636)</td>
<td>4.218 (3.348)</td>
</tr>
<tr>
<td>gray cell</td>
<td>$\nabla$</td>
<td>6.688 (6.544)</td>
<td>1.181 (1.141)</td>
<td>3.811 (4.045)</td>
<td>2.408 (2.355)</td>
</tr>
</tbody>
</table>

Table 2.5: Logic family behaviour at $V_{DD} = 0.3 \text{ V}$ (and $V_{DD} = 1.8 \text{ V}$). All transistors are minimum width (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)

Architectures with fewer stages of gates with higher fanin and fanout may be faster for subthreshold designs, discussed further in Chapter 3. The NOR gates do not do as well indicating the stacked minimum-sized pMOS transistors have a more negative impact at subthreshold than superthreshold voltage.

These results however should be viewed with caution. In Section 5.5.1 on page 109, the latency of larger designs was tested with different process and $P : N$ ratios. There it was discovered that the relative performance between different designs varied as supply voltage was changed. This suggests that the figures for logical
effort also vary with supply voltage in the subthreshold and near-threshold regions. As a result, characterisation of the intended logic library is recommended before hand estimation is undertaken.

2.4 Further Work

A variety of devices, gates and logic families have been proposed, yet attempts at finding an optimal solution are rare.

Although in this chapter the principles of logical effort behaved similarly at subthreshold as they do at superthreshold, later results\(^\ast\) show that this is not necessarily the case. Investigation into the effect of supply voltage on logical effort would be invaluable for it to remain relevant in the subthreshold and near-threshold regions, especially if generalised rules could be determined.

Operating at a subthreshold or near-threshold supply voltage provides the opportunity to revisit prior results. Results determined at superthreshold supply voltages are not necessarily applicable when the supply voltage is lowered. This raises research questions for essentially any architecture or design style that was previously proposed for superthreshold.

2.5 Conclusions

For the 180 nm CMOS process simulated, the behaviour of static CMOS logic gates at subthreshold voltage is not dramatically different to the behaviour one expects at superthreshold voltage – provided process and environment variations are ignored. Absolute switching delays increase exponentially as the supply drops, but switching energy falls quadratically. Static noise margins are well behaved and fall away linearly with the supply level. At a particular operating voltage the linear relationship between fanout and delay is maintained. When normalised against inverter delay at the operating voltage, the slope and no-load intercept of the load line (the logical effort and parasitic delay) are not dissimilar to numbers familiar from superthreshold

\(^\ast\)Section 5.5.1 on page 109
### 2.5 Conclusions

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Subthreshold</th>
<th>Near-threshold</th>
<th>Superthreshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation Delay</td>
<td>– – –</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>Energy Consumption</td>
<td>+++</td>
<td>++</td>
<td>– –</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>+</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>Noise Margin</td>
<td>–</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>Speed as temperature increases</td>
<td>+++</td>
<td>–</td>
<td>– –</td>
</tr>
<tr>
<td>Power as temperature increases</td>
<td>– –</td>
<td>–</td>
<td>●</td>
</tr>
<tr>
<td>Speed as supply voltage increases</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Power as supply voltage increases</td>
<td>– –</td>
<td>– –</td>
<td>– –</td>
</tr>
<tr>
<td>Susceptibility to process variation</td>
<td>– – –</td>
<td>–</td>
<td>+</td>
</tr>
</tbody>
</table>

+ improved/good, – harmed/poor, ● neutral

**Table 2.6:** Comparative merits of gates at different supply voltages

Design. In other words, the relative behaviour of the gate is approximately maintained as $V_{DD}$ drops below the threshold voltage. These and other comparative merits are summarised in Table 2.6.

Experiments with the $P : N$ ratio in an inverter showed that static noise margin could be improved by a small amount by increasing the width of the pMOS FET; however this was at the cost of switching energy and leakage power and provided no significant benefit for average propagation delay. Increasing the length of transistors is a viable method for reducing power consumption at low supply voltages.

Subthreshold behaviour differs in its response to process and environment variations. At subthreshold, propagation delay falls exponentially as temperature is increased, compared to much more gradual increases in delay at superthreshold. The increased variability when operating in the deep subthreshold region is a concern. Either a delay insensitive logic style or large overhead to account for the increased variability is needed. This is discussed in greater detail in Chapter 5.
2.6 Acknowledgement

The material in this chapter is based upon research produced in collaboration with Dr. Braden Phillips and Professor David Blaauw. Some of the text and figures have previously been published in Blaauw, Kitchener and Phillips, *Optimizing addition for sub-threshold logic*, 42nd Asilomar Conference on Signals, Systems and Computers, © 2008 IEEE [10].
Chapter 3

Characterising Subthreshold Adders

The analysis in Chapter 2 was limited to single gates. The principles developed will now be extended to larger designs. The next level of abstraction in digital design is the module. Logic modules consist of multiple gates, typically heterogeneous, combined to perform a particular function.

Unlike the examination of single gates from the previous chapter, it is impossible to exhaustively test the range of possible logic modules. As such, a case study will be used. Adders and carry chains will be explored in detail as these modules are widely used in many different applications and are often on the critical timing path. Larger designs will be examined in Chapter 5.

The approach in this chapter was inspired by the survey of CMOS adders in [118]. The goal in that case was to compare static CMOS and pass-transistor logic styles at nominal (superthreshold) supply voltage, but two aspects of the study strongly influenced the present work. The first was the observation that full adders are not the only important gate for adders and, in fact, most high-performance adders do not use full adders at all. A study of a collection of the most important adder gates including full adders, inverters, NAND, NOR, XOR and XNOR gates, and the and-or-invert and or-and-invert gates used for prefix cells at valency 2 and beyond can be found in the previous chapter.

The second lesson from [118] was the importance of simulation methodology. The authors of [118] debunked many published claims concerning pass-transistor logic made on the basis of erroneous or unrealistic simulation scenarios. Hence, in the present work, care has been taken to ensure that the loading and driving circuits
are realistic, appropriate transitions are observed, environment is specified and other good simulation practices are observed.

Initially attention will be given to the metrics that are important for analysing logic modules. Section 3.2 compares short length (2–9 bit) carry chains. Such chains are used as building-blocks for larger adders as well as other circuits. The critical path of an adder is often the carry path, so by examining the carry chains in isolation relative performance can be estimated. This chapter then proceeds to compare complete adder architectures in Section 3.3.

### 3.1 Metrics

The metrics applied to individual gates are generally applicable to modules. Propagation delay, now often referred to as latency, and power consumption are the two key metrics typically reported. Rather than individual transitions analysed at the gate level, latency can refer to the delay between inputs and outputs of multiple bits, or even of sequences of input and output transitions. Rise and fall time is not considered at this level. There are also new metrics and that will be explored in greater detail: power-delay product/energy per operation and energy-delay product.

#### 3.1.1 Power-Delay Product and Energy per Operation

*Power-delay product* (PDP) is a metric which combines measurements of power consumption and performance to provide a balanced assessment of the circuit’s capabilities. PDP is formed by multiplying the latency, or delay between input and output, by the peak power consumption of the circuit. Modules can easily be optimised for low latency at the cost of greater power consumption and the converse is equally possible. PDP is a metric of efficiency, measuring how well a module performs in both respects. The definition for PDP is shown in Equation (3.1).

\[
PDP = \text{power}_{\text{avg}} \times t_{\text{delay}} \tag{3.1}
\]

Within literature focussing on subthreshold logic, *energy per operation* (EPO) is often used as the primary metric. EPO measures the energy consumed by the circuit for the duration of an action and is defined in Equation (3.2).
Chapter 3 Characterising Subthreshold Adders

\[ EPO = \int_{t_{\text{start}}}^{t_{\text{stop}}} \text{power} \, dt \]  

(3.2)

PDP and EPO are synonyms with the two definitions being equivalent if we take 
\[ t_{\text{delay}} = t_{\text{stop}} - t_{\text{start}} \] and use the integral definition of \( \text{power}_{\text{avg}} \) as shown in Equation (3.3).

\[ PDP = \int_{t_{\text{start}}}^{t_{\text{stop}}} \frac{\text{power}}{t_{\text{stop}} - t_{\text{start}}} \, dt \times (t_{\text{stop}} - t_{\text{start}}) = EPO \]  

(3.3)

As EPO is calculated by the integral of power consumption, measurements generally include leakage power within the calculations. Although static power was found in Section 2.2.3 on page 11 to be unimportant when considering individual gates, it will be shown in Section 3.3.4 that it has much greater importance in assessing the behaviour of subthreshold modules.

### 3.1.2 Energy-Delay Product

Closely related to PDP is energy-delay product (EDP). EDP, as defined in Equation (3.4) is argued by [110] to provide a more realistic metric for architectural performance as it is less vulnerable to manipulation of supply voltage. This claim will be examined further in Section 3.2.3.

\[ EDP = \text{power}_{\text{avg}} \times t_{\text{delay}}^2 \]  

(3.4)

### 3.2 Short Carry Chains

Chains to evaluate a carry across a small number of input bits are a common building-block of larger adders. This section examines the subthreshold performance of different static CMOS chains from 2 to 9-bits long.

The different carry chains are shown in Figure 3.1. The various cells used in these chains, as well as in the adders in the next section, are shown in Figure 3.2. The grey cells use compound (and-or-invert and or-and-invert) gates with fanin 3. These
3.2 Short Carry Chains

![Diagram of carry chains](image)

**Figure 3.1**: 4-bit versions of the carry chains tested: (a) grey cells; (b) inverting grey cells; (c) valency 3 grey cells; (d) valency 3 inverting grey cells; (e) valency 4 inverting grey cells (6-bit).

are extended to valency 3 cells using compound gates with fan-in 5, and to valency 4 cells (not shown) with fanin 7 gates.

Chapter 2 found that NAND gates exhibited good scaling for fan-in at subthreshold, although the opposite was observed for NOR gates. This section explores gates with a mixture of NAND and NOR structures and evaluates the consequence of using high fanin gates, to reduce the number of stages required to achieve the same result.

Papers addressing the opportunities for high fan-in gates at subthreshold are rare. In [10], I compared a valency-3 adder with a normal one. [20] looked into to the limits of subthreshold fan-out and briefly discussed fan-in by claiming the two had
Figure 3.2: Cells used in the carry chains of prefix adders and their equivalent implementations

equivalent models. The analysis was also limited to nMOS transistors, which was justified without explanation that nMOS and pMOS devices are symmetrical. The focus of the paper was on guaranteeing functionality rather than performance or power consumption.

3.2.1 Methodology

The simulation methodology described in Section 2.2.1 on page 9 is also applied here. Spice decks were extracted from layout with interconnect parasitics included. All transistors were minimum width. Unlike the previous chapter, the simulation temperature was set to 70° as [20] associated higher temperatures with poorer reliability. The testbed shown in Figure 3.3 on the next page was used to observe
the rising and falling transitions at the carry output, $C_{out}$ due to a change in the carry input $C_{in}$ when the generate inputs $G_i$ were all 0 and the propagate inputs $P_i$ were all 1. A transient analysis was used to measure the average propagation delay, switching energy and leakage power for these 2 transitions.

### 3.2.2 Measurements

Figure 3.4 shows the average propagation delay at subthreshold and superthreshold supply voltages. Most interesting among these results are the performance of the higher valency grey cells. At superthreshold the chain of valency 3 inverting grey cells is a marginal improvement over the inverting (valency 2) grey cells; the valency 4 inverting grey cells are a step backwards. At subthreshold, the valency 3 inverting grey cells are a significant improvement over the valency 2 cells, and the valency 4 cells provide another small gain. The high fanin cells are offering a greater benefit at subthreshold than at superthreshold voltage.

Switching energy for the chains is shown in Figure 3.5 and leakage power is shown in Figure 3.6 on page 36. The high valency designs demonstrate an advantage in both of these metrics. The leakage results are surprising. At subthreshold the high valency chains are consuming less leakage power than the low valency chains; the opposite occurs at superthreshold voltage.
Figure 3.4: Carry chain average propagation delay from $C_{in}$ to $C_{out}$. Left: $V_{DD} = 0.3$ V Right: $V_{DD} = 1.8$ V

Figure 3.5: Carry chain leakage power. Left: $V_{DD} = 0.3$ V Right: $V_{DD} = 1.8$ V
3.2 Short Carry Chains

Figure 3.6: Carry chain average energy per switch from $C_{in}$ to $C_{out}$. Left: $V_{DD} = 0.3$ V
Right: $V_{DD} = 1.8$ V

Figure 3.7: Leakage power for 6 bit carry chains
To check the behaviour of leakage at subthreshold, a second simulation was conducted using a DC analysis of the valency 2 and valency 4 chains. The average leakage power for the 2 states \((P_6, \ldots, P_1, G_6, \ldots, G_1, C_m) = (1, \ldots, 1, 0, \ldots, 0, 0)\) and \((1, \ldots, 1, 0, \ldots, 0, 1)\) was measured. The results in Figure 3.7 confirm the leakage power for the two designs crosses over between the subthreshold and superthreshold voltages tested. This effect can probably be attributed to the changing contributions of the different leakage mechanisms with supply voltage but further investigation of this result has been left for future work.

Although differences in leakage power have a negligible effect at superthreshold, they become important at subthreshold. In Section 2.2.3 on page 11 it was found that leakage contributed 4% of the total power consumption over a single transition of the gate, effectively measured with an activity ratio of one and measuring no leakage beyond the transition time. For circuits where the activity ratio is significantly less than one or where the cycle time is greater than a single gate delay, leakage power can quickly become the dominant source of power consumption. Choosing gates with the lowest leakage is a viable method to achieve minimum energy operation.

### 3.2.3 EPO and EPD

Combining the measurements of delay, switching energy and leakage provides the metric for energy per operation, shown in Figure 3.8 on the following page. The behaviour at subthreshold and superthreshold is consistent. In both cases the low switching energy of the high valency gates results in reduced values for energy per operation. The magnitude of the changes suggest inverting, high valency gates should be used wherever possible. The largest drops in energy per operation are observed simply by using inverting gates.

To analyse the sensitivity of EPO and EDP to supply voltage, voltage sweeps between 0.3 V and 0.6 V were performed. The results are shown in Figures 3.9 and 3.10. EDP, plotted on a logarithmic graph, displays much more variation compared to EPO. For the grey cell, the EDP at 0.3 V was 27.5 times greater than that at 0.6 V. In contrast to this, the PDP at 0.3 V was only 26% of the value at the larger supply voltage. This behaviour contradicts that described in [110]. As a result, EDP is an inappropriate metric for use at subthreshold, as comparing results measured
3.2 Short Carry Chains

Figure 3.8: Energy per operation. Left: $V_{DD} = 0.3 \text{ V}$ Right: $V_{DD} = 1.8 \text{ V}$

Figure 3.9: Supply voltage vs energy per operation for 6 bit carry chains
Section 2.2.3 on page 11 found that for single gates delay reacted exponentially to variations in supply voltage in the subthreshold region, and this behaviour would continue to apply for larger designs.

3.2.4 Reliability

The low leakage power and PDP of high valency gates at subthreshold are attractive, but these gates are not without their disadvantages.

[20] argues that fan-in has a direct effect on the minimum supply voltage required for correct operation at subthreshold, although the paper concentrates on fan-out, claiming fan-in has an equivalent model. To verify this, a DC analysis was performed to determine the minimum supply required at the typical mean corner. The results are shown in Table 3.1 on the following page. This was determined by measuring the minimum supply voltage at which the output could be driven to a level equal to 90% of the supply voltage. Choosing a lower proportion of the supply voltage is possible, with a value of 50% reducing the minimum supply voltage by approximately 40 mV,
3.2 Short Carry Chains

however a value of 90% was chosen to ensure a more robust noise margin at the edge of functionality. Inverting valency 4 carry chains required a minimum voltage 35% greater than that needed for inverting grey cell chains. Mitigating this by using a higher supply voltage is possible, but it would harm the power savings observed when using high valency gates, especially if high valency gates make up a small proportion of the total design.

<table>
<thead>
<tr>
<th>Chain Type</th>
<th>Fan-in</th>
<th>Minimum Supply Voltage (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverting Grey Cell</td>
<td>3</td>
<td>141</td>
</tr>
<tr>
<td>Inverting Valency 3 Grey Cell</td>
<td>5</td>
<td>174</td>
</tr>
<tr>
<td>Inverting Valency 4 Grey Cell</td>
<td>7</td>
<td>191</td>
</tr>
</tbody>
</table>

**Table 3.1:** Minimum supply voltage for successful operation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean ($\mu$)</th>
<th>Standard Deviation ($\sigma$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMOS Threshold Voltage</td>
<td>100%</td>
<td>6%</td>
</tr>
<tr>
<td>pMOS Threshold Voltage</td>
<td>100%</td>
<td>6%</td>
</tr>
<tr>
<td>Oxide Thickness</td>
<td>100%</td>
<td>3%</td>
</tr>
<tr>
<td>Width Variation</td>
<td>0 nm</td>
<td>9 nm</td>
</tr>
<tr>
<td>Length Variation</td>
<td>0 nm</td>
<td>9 nm</td>
</tr>
</tbody>
</table>

**Table 3.2:** Parameters for Monte Carlo analysis

A Monte Carlo analysis was performed to assess the effect of process variation on minimum supply voltage, as [20] found that it had a significant effect. Variations in threshold voltage for both nMOS and pMOS transistors, oxide thickness, length and width were modelled as Gaussian variables, with the parameters shown in Table 3.2 and a sample size of 1000. The figures provided as percentages are relative to the values specified in the Spice model at the typical mean corner and are an approximation of the influence of process variability on a 180 nm process – the nMOS threshold voltage variation is underestimated, that of the pMOS transistors
are overestimated. The Monte Carlo parameters were pre-generated and the same values were used for all simulations.

<table>
<thead>
<tr>
<th>Chain Type</th>
<th>Minimum Supply Voltage (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverting Grey Cell</td>
<td>144</td>
</tr>
<tr>
<td>Inverting Valency 3 Grey Cell</td>
<td>178</td>
</tr>
<tr>
<td>Inverting Valency 4 Grey Cell</td>
<td>193</td>
</tr>
</tbody>
</table>

Table 3.3: Minimum supply voltage statistics after Monte Carlo simulation

The results of the analysis are shown in Table 3.3. The large standard deviation is to be expected given the significant effect of process variation at subthreshold\(^*\). Histograms for the inverting grey cell and inverting valency 4 cases can be found in Figure 3.11 on the next page. Other than the inverting valency 4 design requiring high minimum supply voltages, the shape of both histograms is similar. At the worst case corner, corresponding to higher voltages required for minimum operation, there is a long tail where increasing the supply voltage provides only a small improvement in reliability. This long tail does not occur in the best case corner, where more favourable process conditions cannot reduce the minimum supply voltage beyond a frequently achieved level.

Care must be taken with these results. If this variation was purely caused by inter-die variation, offending chips could be easily identified at the binning stage. [26] however found that although the distribution of threshold voltage was normally distributed, there was no intra-die or inter-die correlation. In any large subthreshold design, every die will have transistors falling outside of the \(3\sigma\), perhaps 3000 transistors in a million transistor design. This can be mitigated by increasing the supply voltage to a level capable of reliable operation, but this will remove any advantage in power consumption that high valency cells appear to offer.

A method to mitigate the variability of threshold voltages has been proposed by [84], in which the failure of gates to switch appropriately as is modelled as being caused by

\(^*\)See Section 2.2.6 on page 19
3.2 Short Carry Chains

Figure 3.11: Minimum supply voltage required for inverting grey cell carry chain

variations in the threshold voltage of the transistors making up the gate. To solve this, the paper proposes the application of body biasing to offset failure causing threshold voltage variations. This approach is validated through the simulation of the body bias generator and individual gates. Unfortunately, the uncorrelated nature of intra-die process variation would require equipping every gate with an individually adjusted body bias. As this is impractical, any solution would need to divide the circuit into body-bias regions and accept that the approach will fail for a proportion of dies with unfavourable process variation.

The disadvantages of intra-die variation will increase in prominence if smaller process technologies are used, given greater variation in threshold voltage[56]. Because of this, and the approximation of process variation used in the Monte Carlo analysis, these results should be treated as a guide. Before adopting high valency gates or targeting a supply voltage approaching the limits of functionality, repeating these
results is advised in order to verify that the resulting yield is sufficient. The choice of such gates are considered further in the conclusions on page 51.

3.3 Adders

After analysing carry chains in the previous section, attention will be drawn to one of their primary users: adders. Adders are hardware implementations of the mathematical operation of addition. The input consists of two numbers which are summed together to provide the output.

The design of adders is a mature field, having been studied extensively at superthreshold. An overview of the field of adders can be found in [73, 110].

Although full adders are commonly studied at subthreshold to find the optimal design [31, 39] and to examine different transistor types and logic families [15, 41, 96, 104], larger adders have attracted less attention. [70] studied several variants of both 2 and multi operand adders. A modified carry look ahead adder was compared to a conventional one in [29]. [10] compared several Sklansky adder variants against a ripple carry baseline. A bidirectional ripple carry adder was implemented by [19] as part of an asynchronous ALU.

3.3.1 Ripple Carry Adder

The simplest form of adder is the full adder, performing operations on a single pair of bits. By chaining full adders together, the ripple carry adder (RCA), as depicted in Figure 3.12 (a) is created.

Despite the simplicity of the design, optimisations are still possible. The critical path of a ripple carry adder consists of the carry chain. In a RCA comprised of full adders there is an inverter before the output of each full adder stage. By using inverting full adders, shown in Figure 3.12 (b) this can be avoided.

Like the carry chains studied in the previous section, the RCA is a homogeneous design consisting simply of a single repeated gate, so only limited conclusions regarding generic logic can be drawn from it. Even so, it is still useful as a baseline against which other adders can be evaluated.
3.3 Adders

Figure 3.12: 8-bit versions of the adders tested: (a) ripple-carry; (b) inverting ripple-carry; (c) PG ripple; (d) inverting PG ripple; (e) Sklansky; (f) inverting Sklansky; (g) valency 3 Sklansky.
A variant of the RCA is the PG adder, which avoids the use of full adders and is instead constructed out of the cells shown in Figure 3.2. Although the area overhead of the PG adder is larger than the RCA, the critical path delay of the carry chain is reduced. Normal and inverting PG adders are depicted in Figure 3.12 (c) and (d).

### 3.3.2 Tree Adders

The main flaw of ripple carry adders is their slowness. The need for the carry to ripple through each bit of the chain has a direct impact on the speed of the adder. Faster adders, known as tree adders are used where speed is important. Tree adders use multiple levels of logic to increase the speed of the carry path.

The Sklansky adder [94] was selected to represent high-energy, low-delay adders. [75] showed that Sklansky adders can be energy-efficient at supertreshold voltage; and that to optimise their performance it is usually sufficient to place minimum-width transistors everywhere, except for the few high-fanout nodes.

A conventional Sklansky adder is shown in Figure 3.12 (e). Variants of the Sklansky design are also possible. Figure 3.12 (f) depicts an inverting Sklansky adder while a valency-3 variant is shown in (g).

### 3.3.3 Methodology

Once again, the simulation methodology described in Section 2.2.1 on page 9 has been used. Spice decks were extracted from layout with interconnect parasitics included. All transistors are minimum width (8λ), except in the resized Sklansky adders which use either 2-times or 4-times minimum width transistors in the inverters driving the high-fanout nodes on the critical path. Figure 3.13 shows the testbed used to observe both the rising and falling transitions at $C_{out}$ caused by a change in $C_{in}$. The exact transitions used were from $\{A, B, C_{in}\} = \{0\ldots00, 1\ldots11, 0\}$ to $\{1\ldots11, 0\ldots00, 1\}$ and back to $\{0\ldots00, 1\ldots11, 0\}$. This ensures all of the input bits $A_i$ and $B_i$ and the sum bits $S_i$ were toggled to obtain an indication of worst-case switching energy. Simulations were performed at 70°. Transient analysis in HSPICE was then used to measure the average propagation delay, leakage power and switching energy for these transitions.
3.3 Adders

![Adder Diagram]

Figure 3.13: Testbed used for adder measurement (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)

3.3.4 Measurements

Figures 3.14 to 3.16 show the delay, switching energy and leakage power for 8, 16 and 32 bit versions of the adders.

In general, as one would expect, the ripple-carry adders are slower than the Sklansky adders, but consume less switching or leakage energy. The inverting Sklansky adder is slower than the non-inverting version suggesting that fanout has become a problem. There may be scope to improve the former with careful buffer insertion and sizing. Resizing the transistors at the critical nodes of the non-inverting Sklansky adder improves its delay, especially at 32-bits, with little cost in switching energy. The valency 3 Sklansky adder is faster than the valency 2 variant at 8 and 16-bits.

Energy Efficiency

Energy per operation, shown in Figure 3.17 on page 48, tells a different story. The ripple carry adder had the lowest energy per operation, whereas the highest energy per operation was found in the resized Sklansky adders. Interestingly the inverting variants of the PG and Sklansky adders have lower energy per operation than the non-inverting ones, but the opposite applies to the Ripple Carry Adder. From an energy efficiency perspective, the simpler approaches are better. The inverting
Figure 3.14: Adder average propagation delay from $C_{in}$ to $C_{out}$ at $V_{DD} = 0.3$ V (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)

Figure 3.15: Adder average switching energy at $V_{DD} = 0.3$ V (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)
Figure 3.16: Adder leakage power at $V_{DD} = 0.3$ V (from Blaauw, Kitchener and Phillips [10] © 2008 IEEE)

Figure 3.17: Energy per operation at $V_{DD} = 0.3$ V
optimisation for the PG and Sklansky adders work because it removes a significant number of inverters, unlike the inverting ripple carry where there is only a 25% reduction.

As resizing transistors has a negative effect on EPO, it should only be performed to meet timing requirements. The use of resized transistors could also be the reason why the valency 3 adder is not as attractive as the work on carry chains suggests.

An interesting result is that static power ceases to be negligible in the deep sub-threshold region. Figure 3.18 plots the percentage of power attributable to leakage against supply voltage. At 0.3 V static power is responsible for up to 40% of power consumption. The leakage proportion diminishes exponentially until 0.5 V, after which the rate slows. Given the worst case inputs supplied to the adders, the high percentage observed was not caused by a low activity ratio. This analysis assumes that the time interval considered is equal to the latency of each respective adder. If the adders were to be incorporated into a system where a larger period was spent idle, the proportion of energy spent attributable to leakage would be significantly greater.

[10] includes an additional metric describing the net energy saved in a system due to the use of faster adder on the critical path. The fast adder permitted an increased

---

Figure 3.18: Percentage of power attributable to leakage for 32 bit adders
3.4 Further Work

clock frequency and hence saved leakage energy for the entire system. Although it was found that the fast adders reduced total power consumption, the actual amount saved was negligible (up to 100 fW vs the leakage of a single 32 bit adder in the order of nW) hence I conclude that a good rule of thumb for subthreshold systems is to use simple adders such as the RCA.

3.4 Further Work

In the deep subthreshold region, the high proportion of energy consumed due to leakage suggests that system power consumption could be estimated by only examining leakage power. To support this estimate, it could be useful to determine the situations where estimating power consumption only from system size and latency results in an accurate approximation.

Section 3.2.2 found that optimising for leakage power was dependent on supply voltage. Although a hypothesis was proposed that the effect could be attributed to changing contributions of the different leakage mechanisms, no additional study was performed. An investigation into how this works could provide additional insight into how to design subthreshold circuits.

The importance of understanding variability was highlighted in Section 3.2.4 and although some studies have been performed, there are opportunities for further research. One potential approach is to apply the methods used in the modelling of CMOS image sensors to subthreshold logic. The effect of intra-die variability on image sensors has been heavily studied, as it can have a significant impact on image quality. Although some methods to mitigate intra-die variability would not be applicable to arbitrary subthreshold logic (for example post-processing), the models of process variation may be useful in understanding variability at subthreshold supply voltages.

The recommendation for the adoption of ripple carry adders should not be taken as a blanket rule however. Although the case study approach used in this chapter studied the chosen adders and carry chains in detail, the selected architectures may not have been the optimal choices. The use of logic optimisation tools adapted for subthreshold supply voltages would be better placed to analyse competing designs.
and to recommend a result for specific circumstances and optimisation targets, in particular the mutual dependences of latency and power consumption between a given logic block and the remainder of the system.

### 3.5 Conclusion

Although the work on carry chains suggested high valency gates were optimal with low latency and power consumption, they come with the significant cost of reduced reliability in the deep subthreshold region, as summarised in Table 3.4. To maximise yield, the supply voltage supplied should be at least 100 mV greater than the minimum voltage required for the highest valency gates to operate. If the design voltage is below this level, high valency gates should be avoided as the savings attributable to high valency gates will be more than offset by the effect of a higher supply voltage on power consumption or significantly reduced yields.

<table>
<thead>
<tr>
<th>Inverting Logic</th>
<th>Better performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
</tr>
<tr>
<td></td>
<td>Minimal overhead</td>
</tr>
<tr>
<td></td>
<td>Good Reliability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-Inverting High Valency</th>
<th>Gate performance similar to inverting logic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Poor reliability iso-supply voltage</td>
</tr>
<tr>
<td></td>
<td>Higher supply voltage needed for iso-reliability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inverting High Valency</th>
<th>Faster than inverting logic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>More energy efficient iso-supply voltage</td>
</tr>
<tr>
<td></td>
<td>Poor reliability iso-supply voltage</td>
</tr>
<tr>
<td></td>
<td>Higher supply voltage needed for iso-reliability</td>
</tr>
</tbody>
</table>

**Table 3.4:** Summary of the merits of chains of gates compared to low-valency non-inverting logic

Of the various optimisations employed in this chapter, the use of inverting logic has the greatest benefit, improving both latency and power consumption. Although
by themselves inverting high valency gates can offer improved energy efficiency, this advantage is lost when incorporating them into comparatively more complex structures such as prefix trees. Selectively resized transistors can improve latency, but harm energy efficiency.

For adders that are off the critical path, ripple carry adders are the most energy efficient and any additional delay is unimportant. The choice is more nuanced on the critical path. Unless there is a need to meet specific timing requirements, a ripple carry adder is sufficient. Fast adders should only be employed where speed is critical.

This conclusion differs from the one presented in [10] due to different priorities. Here, low power operation is the primary concern and speed is of lesser importance.

3.6 Acknowledgement

The material in this chapter is based upon research produced in collaboration with Dr. Braden Phillips and Professor David Blaauw. Some of the text and figures have previously been published in Blaauw, Kitchener and Phillips, *Optimizing addition for sub-threshold logic*, 42nd Asilomar Conference on Signals, Systems and Computers, © 2008 IEEE [10].
Chapter 4

Architectures for Subthreshold Operation

The approach in Chapters 2 and 3 was to study architecture-independent gates and logic blocks. Several techniques for designing subthreshold architectures will now be addressed. Operating in the subthreshold domain has provided the opportunity to revisit past assumptions and conclusions. In addition to the standard synchronous design practised at superthreshold supply levels, a variety of less conventional approaches have been proposed for use in the subthreshold domain.

Digit serial logic is a methodology where operations are performed on individual digits, rather than words [37]. Such an approach has lower area overhead by significantly reducing the number of interconnects and processing elements at the cost of increased latency, caused by a larger number of pipelining stages. Digit serial logic has been proposed for use at subthreshold in [47] and found to have some promise. This approach would best be used for wakeup circuits in the deep subthreshold region, where leakage is the predominant power consumer and little work is performed.

Dynamic methodologies, such as domino logic [54], have been studied at subthreshold in [28, 96]. Dynamic logic has been found to improve latency and PDP, at the cost of increased switching activity [96]. The additional switching is less of a problem in the deep subthreshold region, where leakage accounts for a greater percentage of power consumption*. As a result, the practice of incorporating a weak keeper results in a substantially greater power consumption when there is a short circuit connection between the supply and ground rails.

*Figure 3.18 on page 49
Two other methodologies will be explored in detail in this chapter: Asynchronous and Globally Asynchronous Locally Synchronous (GALS). A procedure for designing Mixed Timing Domain (MTD) systems, of which GALS is an example, is proposed and is applied to expose timing errors in published designs.

### 4.1 Motivation

Once the decision to operate at subthreshold or near-threshold supply levels has been made, there are further decisions on how the system should be designed. Many architectures designed for superthreshold supply levels will continue to function when the voltage is reduced, but two key questions must be asked. Is this an efficient use of a severely limited power budget? Does the architecture take into account the advantages and disadvantages of operation at subthreshold supply levels?

In choosing an architecture for use at subthreshold, there are two primary considerations. Firstly it should be energy efficient. The main motivation for using a subthreshold supply is to save power. This should not be hindered by the choice of an inefficient architecture. The second consideration is the severe process variability observed when operating at subthreshold\(^*\). A good subthreshold architecture would mitigate the effects of variability, especially intra-die variation. These goals must be balanced against the ease of development. An architecture with optimal energy efficiency and variation tolerance is of limited use if it is impractical to design systems based on that architecture.

### 4.2 Asynchronous Design

An asynchronous design is one where there is no clock signal. Instead information can be processed or transmitted between modules at any time. Reliable communication between modules is achieved using handshaking. A transaction is requested by the sender and is not completed until acknowledged by the receiver. Overviews of asynchronous logic can be found in [38, 67, 97]. There are a variety of possible asynchronous design styles, which can be classified in three broad ways.

\(^*\)Section 2.2.6 on page 19
Chapter 4  Architectures for Subthreshold Operation

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Neutral</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Logical 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Logical 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal State</td>
</tr>
</tbody>
</table>

Table 4.1: Data representation in dual rail designs

Data Representation

The first way to classify asynchronous systems is based on how the data is represented. Single rail designs use one wire per bit whereas a dual rail implementation uses two wires per bit [67]: one wire transfers a logical 0 and the other represents a logical 1, as shown in Table 4.1. This representation allows a state to indicate neither bit, used when no information is being transmitted. This can also be generalised to $N$ wires and $N$ possible values in an encoding known as one-hot. Although twice as many wires are required to transmit a byte of data, systems can be constructed with fewer signal wires, which represent the state of the system.

Completion Detection

Secondly, asynchronous systems can be classified by how the system knows when processing is complete. This can simply be accomplished with dual-rail systems where one of the possible states indicates that the operation has not yet been completed. Completion is signalled when all outputs report a logical 0 or 1 state. The advantage of dual rail designs is that completion detection is possible without any additional logic due to the inclusion of the neutral state. This cannot be relied upon for single rail designs, where a variety of techniques have been proposed. The simplest method is bundled delay line [13]. Bundled delay works by duplicating the latency of the critical path with a delay line constructed using a chain of gates. The input to the delay line is toggled when the operation begins so that a transition at the output indicates that sufficient time has elapsed for the operation to have
4.2 Asynchronous Design

completed. Bundled delay requires careful attention when designing systems as the critical path must be correctly selected to avoid timing failures.

Completion detection can also be performed through current sensing [32]. An amplifier is connected between the power rail and the asynchronous module. Switching activity is observed via an elevated current reading and completion is measured by the return to a steady state current level. This approach is more reliable to design than bundled delay as the possibility of choosing an incorrect critical path is removed. It does however suffer from the disadvantage of including analogue components within a digital design, which require very different design skills and make porting the circuit to another process technology more difficult.

Handshaking

The third classification relates to how modules communicate, or perform handshakes, with each other. Handshakes can be performed in two principle ways. Two phase handshakes use two patterns, alternating between them for each transaction. These are shown in Equation (4.1). R represents request, a signal controlled by the sender, and A corresponds to the acknowledge signal, governed by the receiver. For the first transaction (and subsequent odd numbered transactions), the sender raises R at the beginning of the transaction and waits for the receiver to raise A to acknowledge the transaction has been completed. For the second (and subsequent even numbered transactions), the sender lowers R and waits until the receiver lowers A.

\[
\begin{align*}
R & \uparrow A \uparrow \\
R & \downarrow A \downarrow 
\end{align*}
\]  

(4.1)

Four phase handshakes follow a pattern described in Equation (4.2). Like the first two phase handshakes, the transaction commences when the sender raises R, but the subsequent steps differ. When the receiver raises A, this indicates that the receiver has finished the processing required for the transaction. The transaction only completes when the sender lowers R and the receiver acknowledges this by lowering A. This pattern restores the logic to its initial state at the conclusion of the transaction by combining the two separate handshakes from Equation (4.1) into one action. Compared to two phase handshakes, latency is increased by the need to return to zero, but the logic required to implement the handshake is simplified.
One key disadvantage of asynchronous design relates to testing. Testing of conventional synchronous designs can be split into timing and functional verification [85].

Timing verification involves finding the longest and shortest delay paths within a circuit. This is then used to ensure the frequency and hold time specifications are met and as a starting point for optimisations to reduce the delay. With synchronous systems, the clock transition can be used as the reference as all signals change relative to the clock signal. This is not possible for asynchronous systems where there is no central clock and signals can transition at any time.

Functional verification involves ensuring that a system will produce the correct answer for a given set of inputs. This is easily accomplished for individual components, but is more difficult for inter-related modules as loops can form, leading to infinite recursion. Synchronous verification can use the presence of clocked registers to escape such loops [85], but this technique cannot be relied upon with asynchronous verification.

In asynchronous design it is not possible to separate timing and functional verification as correct function relies on the relative timing of signals, increasing the complexity of verification efforts.

In addition to the problems with verification of asynchronous systems, the tools and techniques used for verifying such designs are less mature than those designed for synchronous systems. Several methodologies for asynchronous verification have been proposed [9, 88, 112], but tools based upon such methods have not undergone the rigorous testing and years of commercial development of conventional synchronous tools.

### 4.2.1 Asynchronous at Subthreshold

Asynchronous design is the most heavily studied non-traditional methodology for subthreshold and near-threshold circuits, and for good reason. Such methodologies
are well suited to subthreshold operation as they provide both energy efficiency and robustness to process variation.

Asynchronous designs achieve energy efficiency by eliminating the clock signal. Circuits only need to transition when there is work to be performed. [43] justifies an asynchronous architecture by citing this motivation. That asynchronous design can make circuits more robust to inter-die and intra-die variability, is argued by [13]. Although inter-die variation can affect the performance or power consumption characteristics between chips, intra-die variation is caused by non-uniform variations and means that the critical path in a circuit may vary between chips. This will make isolating the cause of timing failures problematic. Choosing a slower clock frequency is often a solution, but the variability observed in Section 2.2.6 on page 19 would require a clock that is significantly slower than ideal for typical conditions and the additional static power consumption would harm efficiency and battery life in the deep subthreshold region. An asynchronous methodology solves this by letting every part of the design work at its own pace.

Of the various techniques used in the design of asynchronous systems, many have been applied to subthreshold designs. [13] proposes a bundled delay methodology to account for both inter-die and intra-die variations. This is accomplished by duplicating the delay lines. [90] uses a simpler bundled delay method for memory write completion detection, in an otherwise synchronous design. Current sensing is performed by [1] at subthreshold.

Null Convention logic (NCL) is a proprietary technique proposed by [27]. It is a quasi-delay-insensitive dual rail scheme based on threshold gates. [43] uses NCL in the design of a microcontroller for subthreshold operation.

[64] studies and compares a variety of these techniques, highlighting the potential of dual rail designs.

4.3 Mixed Timing Domain Systems

There are situations where an all-asynchronous design is impractical. The reuse of existing components, or the licensing of third party modules are common practices.
As the dominant design methodology is synchronous, this limits the ability to incorporate these components into an asynchronous design. Although these modules could be redesigned to adopt an asynchronous methodology, this incurs significant development overhead and is not always possible for third party designs. The increased verification cost incurred by asynchronous systems may also make the design infeasible.

Alternatively, a large synchronous design may have difficulty satisfying all timing constraints with a single clock. The critical path of a third party component might be too long to support a target clock rate required by another module. There may be large area and power overheads involved in distributing the clock across the entire chip while avoiding clock skew.

A solution to these problems is to adopt a Mixed Timing Domain (MTD) approach. A timing domain consists of a set of modules that share a common timing methodology. In a synchronous system, this would be a single clock frequency. In an asynchronous system, the common element would be the completion detection mechanism. An MTD system consists of more than one timing domain. These timing domains could be synchronous, with each domain having a different clock signal, or there could be a mixture of asynchronous and synchronous domains.

An example of a MTD system is shown in Figure 4.1 on the following page. Depicted are two modules with independent clock signals and a third module implemented using an asynchronous methodology. Communication between these domains is handled by specially designed components called timing interfaces (TI). These are described further in Section 4.4 and example implementations are provided in Chapter 5.

### 4.3.1 Globally Asynchronous Locally Synchronous Systems

One possible implementation of MTD methodologies is *Globally Asynchronous, Locally Synchronous* (GALS) [55]. GALS works by splitting a design into multiple independent, synchronous timing islands, each with its own clock. Where islands communicate, they do so using asynchronous interconnect. This removes much of the overhead of a global clock distribution tree, at the cost of increased latency.
4.3 Mixed Timing Domain Systems

Figure 4.1: A Mixed Timing Domain system consisting of two different clock domains

Figure 4.2: A GALS system with multiple synchronous domains communicating over an asynchronous channel

when communicating between islands. Figure 4.2 depicts a GALS system with synchronous modules communicating over an asynchronous channel. Point to point links are also possible between two different synchronous domains.
Chapter 4 Architectures for Subthreshold Operation

The alternative to generating multiple clock frequencies or using asynchronous components is the use of a single frequency clocked fast enough to meet performance requirements. Although judicious use of clock gating may mitigate the energy use of a faster than necessary single clock, it does not address any overhead incurred by meeting the timing requirements, nor does it mitigate the expense of a clock distribution tree.

4.3.2 GALS at Subthreshold

Subthreshold or near threshold systems are designed with the primary goal of minimum energy operation [107]. Lowest power operation can be achieved by splitting the design into multiple power regions and clock domains [117]. A GALS methodology is ideal for efficient communication between these regions.

In addition to the gate level analysis described in Chapter 2, there have been several substantial designs targeting the subthreshold and near threshold domains. The Phoenix processor [34, 91] and the Intel near-threshold x86 processor [42] are two examples that have been fabricated. [43] reports simulations of a microcontroller. What is lacking in literature is an examination of how large heterogeneous designs are connected together. This chapter proposes the use of GALS [55] in the subthreshold or near threshold domains where energy usage would be minimised by running each component at an optimal, individual frequency. The oscillators presented in [59, 62] and [63] could be used to provide locally generated clock signals.

The motivations for GALS at superthreshold are equally valid in the sub or near threshold domains. Moving to an all asynchronous design may not be possible if third party IP blocks are involved, and the inefficiencies of a single clock are more pronounced when minimising power consumption is the primary objective.

4.3.3 Mixed Timing Domain Topologies

In this section, some example topologies are provided and qualitatively compared. Although GALS is often targeted at network on chip (NOC) applications [8, 101, 113] it is proposed that for subthreshold systems, the principles be applied at a smaller scale. Rather than packet switched mesh networks, simpler designs such as
4.3 Mixed Timing Domain Systems

Figure 4.3: Point to point ring topology

Point to point connections or a ring bus may more suitable for a cheap, lightweight interconnect.

Figure 4.2 on page 60 depicts a traditional GALS architecture. Each timing domain has an independent clock, which can be turned off when the domain is powered down. This thesis concentrates on the conversion between timing domains. As such, the exact nature of the asynchronous interconnect is unimportant.

Point to point links connect two synchronous domains directly, without the use of an intermediate asynchronous layer. These can be incorporated if two timing domains are tightly related with little communication to other domains. The power and latency saved by avoiding an additional conversion to/from the asynchronous domain needs to be weighed against the overhead of providing multiple timing interfaces.

An alternative to the conventional approach is the sole use of point to point links, as depicted in Figure 4.3. The asynchronous interconnect is removed and each synchronous domain talks directly with its neighbour. This approach is best suited for small designs, where the number of timing domains is low. While the bulk of a
clock domain can be shut off, the timing interfaces and clock generators must remain active.

While this approach would have reduced area requirements, the latency of the system would be larger given the greater number of timing domain transitions.

A further disadvantage of the point to point ring is that it enforces severe constraints on the performance of the system if one domain is clocked significantly slower than the others. This can be solved by segregating the slow clocks into a secondary ring.

4.4 Mixed Timing Domain Design

Section 4.3 introduced a mixed timing (MTD) domain methodology but did not describe how such systems were designed. This section details a methodology useful in designing MTD systems.

The foremost challenge in designing the timing interface between MTD systems is minimising the effect of timing failures. Any time a signal reaches a timing element, such as a flop, there is a possibility of an error in transmission if the timing constraints of the receiving timing element are violated.

Figure 4.4 displays the principle timing constraints of a D-flop. The signal $D$ represents the input to the flop. For the majority of the clock cycle, $D$ is permitted to be arbitrary, changing to any state at any time. There are however two constraints
4.4 Mixed Timing Domain Design

associated with the rising edge of the clock. Setup time requires the input to remain constant for a sufficiently long interval before the rising clock transition. This can be resolved by reducing either the critical path or the frequency of operation. Hold time requires the input to the flop to remain constant for an interval after the rising clock transition. Increasing the time required for the signal to propagate from the previous stage can resolve hold time violations.

The outcome of a timing violation is unpredictable. It can result in an incorrect signal being read, or it could cause the system to behave non-deterministically in a manner called metastability [30]. The possibility of errors can be eliminated by ensuring that all timing constraints are satisfied for an input that occurs at a given time. For single domain systems this is easily satisfied, as the start is constrained by the previous clock cycle [85]. This cannot be applied generally to MTD systems as there are fewer constraints on when the input can change.

Designs must also be tolerant to a wide range of operating conditions. Process variation, especially intra-die variation, is much more significant in the subthreshold regime. Theoretical analysis of timing delays cannot guarantee correctness if the fabricated delays are non-constant and simulations rarely include multiple simultaneous design corners. Delay insensitive asynchronous designs [38] can be used to compensate for this high variability, but this is not always practical.

Mixed timing domain systems are hard to design correctly, as timing failures are easy to introduce and hard to debug. The key is to identify the exact interface between two timing domains and to then ensure that all signals passing through are protected.

4.4.1 Methods to Prevent Timing Failures

Within the timing interface, circuit modules must be used to ensure the conversion between timing domains is performed safely. These modules are generically referred to as synchronisation elements, even when applied to asynchronous domains.

**Synchronisers**

Synchronisers are commonly used to ensure safety. Synchronisers are implemented by a sequence of D-flops. They do not completely eliminate the possibility of timing
failure. Instead their purpose is to make timing failures sufficiently improbable to occur in practice. The probability of failure is a function of both synchroniser length and clock frequency. To lower the probability, either the synchroniser length can be increased (causing a corresponding increase in latency), or the frequency of operation can be reduced. [6, 111] provide methodologies to determine the probability of failure. Based on these methodologies, tables providing the chance of metastability can be found in [7, 72].

The disadvantage of synchronisers is that they require the data transmitted to be delayed for a period of time. Systems must be designed to accommodate the additional latency imposed by synchronisers. Consider a signal that indicates whether a buffer is full, to prevent overflow errors. If this crosses timing domains, overflow may occur while the transition to a full state is stabilised by the synchroniser. This situation can be avoided by triggering the full state earlier, before the buffer is actually full.

Clock Pausing

An alternative is to temporarily halt the destination clock when information is transferred, in a technique known as clock pausing [114]. The clock is stopped when information is transferred across the timing interface.

Clock Relationships

Timing failures can be avoided by imposing additional constraints on the relationship between the clocks on either side of the interface. Taxonomies of the relationship between clock signals can be found in [99]. For example, consider a system with two clock signals where the frequency of one clock is an integer multiple of the other clock. The two clocks can then be described as ratiochronous. When transmitting data from the faster clock domain to the slower domain, no action needs to be taken as the signal will automatically meet the timing constraints of the slow clock, providing the signal meets the timing constraints of the faster clock and the setup and hold times of both domains are identical.

When transitioning from the slower clock to the faster clock, there are now multiple setup and hold time intervals. The system is guaranteed to be safe if each of the
setup and hold constraints are met. The simplest method is to ensure that the logic in the transition region satisfies both clock constraints by bounding the critical path to the minimum of the two permissible limits. It is possible to incorporate logic with a critical path greater than the minimum period, at the cost of significantly increased verification effort. Unlike the method described in [85, Chapter 5], it is not sufficient to select the path with longest delay as glitching may violate the constraints. Instead, all possible combinations of inputs must be checked against the timing constraints. Minimising the logic within the timing interface is necessary to avoid a large combinational state space.

**Guarding**

A signal that fails to meet timing constraints can be combined with a signal that meets those constraints if and only if the first signal is guaranteed not to change state (even momentarily) while the second is active. This is called guarding and is illustrated in Figure 4.5. The message signal, $\text{Data}_1$, and the guard signal, $\text{Safe\_guard}$ initially satisfy the timing constraints of timing domain 1 whereas $\text{Data}_2$ belongs to the second clock domain. When $\text{Safe\_guard}$’s low state is passed through a two flop synchroniser, it has the effect of forcing $\text{Data}_2$ to a constant state which is independent of any changes in $\text{Data}_1$. 

![Figure 4.5: Implementation of clock guarding](image-url)
When timing domain 1 can guarantee that the value of $\text{Data}_1$ will not change, it can set $\text{Safe\_guard}$ high. This signal is then passed through the synchroniser into clock domain 2. This signal is then combined with $\text{Data}_1$ to create $\text{Data}_2$ which belongs to clock domain 2. Throughout this process, domain 1 needs to be aware of when domain 2’s instance of $\text{Safe\_guard}$ is high to ensure that it does not change $\text{Data}_1$. This could be performed through a handshaking mechanism or knowledge of the second domain’s clock frequency.

Any communication between the two systems to notify the receiver of such a state still needs to apply a different method of resolving timing hazards, but applying guarding in addition to that method has the potential of reducing area and power overhead when the guarded signal is a bus. For example, to apply two stage synchronisers to each element of a 32 bit bus would require at 64 flops in total. This can be reduced to as little as 2 flops and 32 NAND gates by guarding the bus and applying a synchroniser to the guard signal.

Tristate inverters and multiplexors can also be used in addition to NAND gates. When using tristates, care needs to be taken that output is not left floating as noise on the floating signal has the potential to cause a timing violation. Guarding is suitable for systems consisting of multiple stages if read and write access occurs on different modules. This is implemented for the output bus in the FIFOs described in [17] and Chapter 5.

**Glitch Prevention**

Glitches, or temporary changes in output between the initial value and the final computation of a circuit, are typically considered unimportant in synchronous systems. However in the field of asynchronous design, glitches are referred to as hazards as they have the ability to cause incorrect operation of a circuit. Spurious values can be misinterpreted as deliberate transitions by subsequent stages.

Asynchronous circuits (and synchronous circuits with inputs into asynchronous systems) can be redesigned to avoid the possibility of hazards. One method to accomplish this is to add redundant states or logic to a circuit. Further information on how to classify and resolve hazards can be found in [44, 102].
4.4 Mixed Timing Domain Design

4.4.2 Conversion between Synchronous and Asynchronous Domains

Transitioning from a Synchronous to an Asynchronous domain is more difficult. Although there is no chance of metastability, timing hazards are still problematic and the logic at the interface needs to conform to asynchronous design styles. The prevention of glitching is the highest priority. Without the benefit of a handshake protocol, the logic within the interface needs to ensure the signal stays high long enough for the asynchronous design to properly process it. An example of a synchronous to asynchronous converter is described in Section 5.4 on page 106.

4.4.3 Procedure for Designing MTD Systems

After identifying the timing interface, there is a need to account for all signals passing through it. These interfaces should be kept as small as possible to minimise the design and verification effort. The following procedure is proposed to identify and resolve these interfaces. For clarity, a worked example is provided based upon
the system depicted in Figure 4.6. The system has two input clocks and contains some asynchronous logic. The wires which will be referred to in the example have been specifically marked.
1. Assign all external signals to and from the system to a domain. These domains are called external domains. There is one external domain for each clock input and one for each style of asynchronous logic. If a module accepts more than one clock input, it needs to be subdivided into multiple domains.

*There is one asynchronous external domain, EA1 (brown), and two external synchronous domains ES1 (blue) and ES2 (green), one for each of the clock inputs, clk1 and clk2. Module 2 has been divided into two parts, 2a and 2b because it was originally drawn accepting both of the clock inputs.*
2. Assign each module to an external domain if it has inputs from exactly one external domain, repeating this step until no additional domains are marked.

Initially module 1 is assigned to domain ES1, 6 to domain ES2 and 3 to EA1. Module 2a can be assigned to domain ES1 as module 1 also belongs to ES1. These have been shaded in the colour of their respective domain. Modules that accept inputs from unassigned domains are left unmarked.
3. Assign a new domain for each module that accepts inputs from two or more external domains. These are referred to as clash domains. Affected modules are marked as clash domains in addition to their existing status as external domains. There is one clash domain for each combination of external domains.

4. Repeat steps 2 and 3 until all modules have been assigned to a domain. If unassigned modules remain, assign one to a new external domain and repeat the process.

*Modules 5 and 2b accept inputs from domains ES1 and ES2 and are assigned to clash domain C1 (shaded red). Module 4 has inputs from domains EA1 and C1, so it is assigned to domain C2 (orange). If there were no asynchronous domain inputs to module 4, it would be instead assigned to domain C1. Although
it has an additional input from domain ES2, as ES2 is one of the domains involved in the creation of C1 it belongs to the same domain.
4.4 Mixed Timing Domain Design

5. Identify which clash modules must belong to an external domain. These are modules that accept a clock signal from an external domain or have output signals that leave the system belonging to at most one external domain. This is referred to as forcing the domain.

*Figure 4.10:* Procedure step 5

*Modules 2b and 4 must belong to domain ES2 as they accept a clock input and have outputs belonging to that domain. To indicate their forced status, the modules have been partially shaded in the colour of ES2 (green).*
6. Highlight the signals where the destination domain differs from the origin domain. Do not mark signals that originate in a forced domain if the destination belongs to the same domain.

*These signals have been indicated on the diagram by the use of a bold arrows. Signals w4, w7 and clk2 are not highlighted as domains 2b and 4 have both been forced to belong to domain ES2.*
7. Starting from the transitions from external domains into clash domains, apply synchronisation elements to the signals identified in step 7. The synchronisation elements will resolve the clash so that the module will belong to one of the external domains.

A synchronisation element has been used on signal w8, indicated on the diagram as a black rectangle. There is a choice in resolving module 5’s clash status. The synchronisation element can be applied to either w1 or w5. If the element was applied to signal w1, module 6 would subsequently belong to domain ES2. Likewise it would belong to ES1 if the synchronisation element was applied to signal w5. In this instance, it is more efficient to apply the process to signal w1 as it will reduce the total number of synchronisation elements required.
8. Repeat steps 2–7 until all clash modules have been eliminated.

A synchroniser is applied to Signal w2, resolving module 4’s status as a clash domain. If the synchronisation element was applied to w5 instead of w1 in the previous step, an additional synchronisation element would be required for w3.
9. Apply synchronisation elements to the remaining signals identified in step 6.

The only remaining identified signal is w6 and this can be resolved by applying a synchronisation element. The system is now safe from timing hazards.

Figure 4.14: Procedure step 9
Once clash domains are identified, they need to be examined to determine their behaviour. If the domain has internal state or memory but no dependence on a clock signal, it is an asynchronous domain. It is important to recognise this as the constraints of asynchronous design, notably the intolerance to glitching, will affect the design of the surrounding components.

Synchronisation elements do not need to be placed directly on the interface between external and clash domains. Placing them further into the clash domain is possible for modules which consist entirely of combinational inputs and have a synchronous destination, especially if it reduces area or delay overhead. Suppose a combinational module accepts three inputs from timing domain A and one from clock domain B, with a single output connected to clock domain B. If the synchronisation elements are placed at the inputs to the module, three elements on domain A’s inputs would be required. However, as the module consists of purely combinational logic, the synchronisation element can instead be placed at the output, requiring only one synchronisation element. If instead domain B was an asynchronous domain, this will not be sufficient as it has the potential for glitching. The module would need to be designed to eliminate this possibility.

4.4.4 Limitations

Although the proposed approach can reduce the occurrence of timing failures, it cannot completely eliminate them. All flops have setup and hold time constraints, the violation of which can cause non-deterministic behaviour. No matter how finely a synchronous system is partitioned, there is always the potential for metastability induced by arbitrary process variation. Asynchronous designs suffer from similar problems. Although delay-insensitive designs can completely eliminate timing assumptions, only a small subset of available designs can be implemented using them [68]. As a result, most practical implementations assume additional requirements, with quasi-delay-insensitive requiring the fewest constraints. QDI systems rely on what are termed isochronic forks. These are branches that satisfy the following condition:
4.4 Mixed Timing Domain Design

“In an isochronic fork, when a transition on one output is acknowledged, and thus completed, the transitions on all outputs are acknowledged and thus completed.” [68]

Martin [68] then states the timing assumptions that the isochronic fork requires. The first assumption is that the delay in each branch of the fork can be considered equivalent. Ignoring the presence of inverters (these can be removed by circuit transformations), this is trivial to demonstrate at subthreshold as wire propagation delay does not vary with respect to supply voltage.

Secondly the switching threshold of the fork’s output gates are sufficiently close. As subthreshold designs inherently suffer from high process variability*, this assumption cannot hold once intra-die variation is considered. Hence formal correctness cannot be proved.

Although in each case process variation can prevent 100% yields from being achieved, careful design and the non-uniformly distributed nature of process variation can allow an acceptable level of functional chips, even in the high variability environment of subthreshold operation.

In addition to the problems of process variability, not all systems can take advantage of mixed timing domain architectures as the approach of letting subcomponents operate at their own pace relies on a degree of independence in the interaction of the components. An example of a system that cannot take advantage of this methodology is that of a linear pipeline. Although individual timing domains may be capable of processing data at a faster rate due to more favourable process variation, the domains and the system as a whole are limited to the pace of the timing domain with the highest latency.

4.4.5 Application to Published Designs

By applying this procedure to previously published designs several timing safety flaws have been uncovered.

The examples provided are all implementations of timing interfaces, designed with the goal of safely transmitting information between timing domains. The occurrence

*See Section 2.2.6 on page 19
of such flaws in these designs highlights the need for a methodology to prevent timing problems.

Each design implements a FIFO (First In First Out) interface, transmitting information between timing domains in the order in which it is received. Further discussion of FIFOs, including these examples, can be found in Chapter 5.

Due to the limited availability of subthreshold designs which have been described in sufficient detail, the example systems are those which have been designed to operate at superthreshold supply voltages. Superthreshold designs do not suffer from the severe process variability that is inherent in subthreshold operation, allowing more timing assumptions to be made. As I am applying stresses to which the designs were not expected to be exposed, the discussed problems may not be applicable to their intended operational environment.

**Problem in Chelcea and Nowick’s Design [17]**

The Chelcea design implements a family of FIFOs capable of transferring data between the possible combinations of synchronous and asynchronous timing domains. The diagram for a mixed clock (synchronous to synchronous) FIFO stage is reproduced in Figure 4.15. The stage can be initially divided into a put side which transmits data and a get side that receives data. This leaves the Data Validity Controller (DVC), which determines whether the stage is full or empty. The DVC belongs to a clash domain as it accepts data from both put and get domains. For the mixed clock FIFO it is implemented with an SR flop. As the implementation has internal state, no clock inputs and an intolerance to glitching, the clash domain status can be partially resolved by assigning it to a new asynchronous domain.

This action requires that the inputs to the DVC do not glitch. Should \texttt{en.put} be momentarily high, this could trigger a change in the DVC state from empty to full. As \texttt{en.put} does not remain high, no data will be written to the stage’s register and the token will not be passed to the next stage. As the put side only checks whether the FIFO is full, it will still believe the FIFO stage is empty and ready for writing.

After accounting for synchroniser delay, the get side will then believe the FIFO stage is both full and safe for reading. The most likely outcome is that the get side will simply read the contents of a previous message. Should the put side decide to
4.4 Mixed Timing Domain Design

Figure 4.15: Mixed-clock FIFO stage from the Chelcea design. Diagram adapted from Fig. 6, [17] © 2004 IEEE

write to the FIFO stage at this point, it could change the contents of the registers while they are being read, with the potential to cause metastability as the changes will flow through the tristate buffer unhindered. If both \texttt{en\_put} and \texttt{en\_get} are high, the DVC will be forced into an illegal state, with an unpredictable outcome. Similar behaviour will occur should the get side momentarily trigger \texttt{en\_get}. The Data Validity Controllers for the synchronous to asynchronous and asynchronous to synchronous variants are implemented differently, but still suffer from the same flaw.

To fully resolve this problem, the logic controlling the inputs to the DVC should be required to conform to asynchronous design considerations to remove the possibility of glitches. This can be done by shielding \texttt{req\_put} and \texttt{req\_get} with edge triggered D-type flops, the implementation of which may have been assumed by the authors.
The FIFO presented in [72] approaches the problem in a similar way. As in Chelcea, the Ono synchronous to synchronous FIFO can be divided into three modules: the Put Interface, the Get Interface and the Full/Empty Controller (FEC). Similar to the Chelcea DVC, The FEC is also implemented using an SR flop, however additional logic prevents the possibility of entering an illegal state where both inputs to the controller are high.

The proposed method was applied to the synchronous to synchronous FIFO stage, as shown in Figure 4.16. The FEC can be assigned to an asynchronous domain, with the put and get logic assigned to put and get domains respectively. However part of
4.4 Mixed Timing Domain Design

the FIFO stage is still flagged as a clash domain. The tristate buffer∗ accepts input from a latch belonging to the put domain as well as the signal \texttt{get\_token\_in} which belongs to the get domain. When the FIFO is empty, both the put and get tokens are present for the same stage. When the put stage writes data into the latch, it is passed immediately through to the tristate bus. As the stage holds the put token, the control signal is high and the written data flows through to the \texttt{dataout} register†. This places both the tristate and the flop in the clash domain, leaving the flop vulnerable to timing failures.

There are two ways to resolve this flaw. One way is to replace the output flop with a synchroniser, but this will cause a significant increase in latency. An improved method would be to define the tristate buffer as a timing interface, and use a more appropriate guard signal. Tristate buffers are suitable as timing interfaces as they can prevent inopportune glitches from propagating to a subsequent stage if it can be guaranteed that the input to the tristate cannot switch when an enable signal is high. A further condition is that the enable signal cannot cause timing violations, typically by belonging to the same timing domain as the input to the subsequent stage. In Ono’s design, the \texttt{read} signal is a suitable candidate to act as a enable signal. This signal will not be raised until after the latch has been set and an appropriate synchroniser delay has elapsed. While it is high, the latches of that FIFO stage will not be written to as the put interface will still classify the FIFO stage as full. In this way the clash domain is resolved and can be reassigned to the get domain.

Problem in Rahimian and Mohammadi’s Design [79]

Rahimian and Mohammadi use a different approach in designing FIFOs. Rather than providing a family of FIFOs to address the possible combinations of synchronous or asynchronous logic, a single FIFO design is used with synchronous or asynchronous wrappers at either end of the FIFO to interact with the target system. This separation is not complete however as the central FIFO still has sections belonging to the two outer timing domains.

∗The Ono design may have instead used a transmission gate here, but the behaviour is equivalent
†[72] Figure 10
[79] claims that the use of RxPermit and TxPermit eliminates the need for synchronisers in their FIFO. It is these very signals that can introduce timing hazards. Figure 4.17 depicts the problem in the full case. HeadAddr is the output of a flop controlled by the signal put, which exists in the put (Tx) domain. TailAddr likewise exists in the get domain (Rx). Therefore the output of the comparator, full, belongs to a clash domain. The attempt at guarding the Permit signal with the TxClock is not sufficient as it still leaves it vulnerable to setup time violations as TailAddr could change at any time.

## 4.5 Further Work

That errors that have been identified in published work demonstrate both the difficulty of designing correct Mixed Timing Domain systems and the need for a timing verification methodology such as the one described in Section 4.4.3. There is an opportunity for further research to automate this process and to improve it by identifying optimal locations to insert timing interfaces.

The proposed methodology treats a design at the block diagram level, but there are other levels of abstraction that it does not address. Applying the methodology to a design can put constraints on the architectural and physical partitioning of designs, but the consequences of this have not yet been evaluated. One of the questions that could be answered is whether there is an upper limit to the size of a timing domain,
### 4.6 Conclusion

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronous</strong></td>
<td>High availability of third party IP</td>
</tr>
<tr>
<td></td>
<td>Mature, well developed design tools</td>
</tr>
<tr>
<td></td>
<td>Mature, well developed verification tools</td>
</tr>
<tr>
<td></td>
<td>Vulnerable to process variation</td>
</tr>
<tr>
<td></td>
<td>Best for ease of design and verification</td>
</tr>
<tr>
<td><strong>Asynchronous</strong></td>
<td>Limited availability of third party IP blocks</td>
</tr>
<tr>
<td></td>
<td>Resistant to process variation when using dual rail methodologies</td>
</tr>
<tr>
<td></td>
<td>Verification difficult – cannot separate timing and formal verification</td>
</tr>
<tr>
<td></td>
<td>CAD tools less capable and mature.</td>
</tr>
<tr>
<td></td>
<td>Best for reliability</td>
</tr>
<tr>
<td><strong>Mixed Timing Domain &amp; GALS</strong></td>
<td>Compromise approach between synchronous and asynchronous</td>
</tr>
<tr>
<td></td>
<td>Allows incorporation of third party IP</td>
</tr>
<tr>
<td></td>
<td>Effects of process variation on synchronous components reduced</td>
</tr>
<tr>
<td></td>
<td>Partitioning of design simplifies verification efforts</td>
</tr>
<tr>
<td></td>
<td>Individual clocks and supply voltages can improve energy efficiency</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of high-level design methodologies when applied to subthreshold designs

or the circuits abstracted as logic blocks. In the case of a synchronous design, the larger the domain the more likely that intra-die variability will have a detrimental effect on performance. However, is there an optimal balance between the lower performance of larger domains and the additional overhead of timing interfaces?
4.6 Conclusion

At first glance, synchronous design methodologies seem to be the most effective option for subthreshold design. The tools for design and verification are more capable, third party IP is readily available to reduce design costs and the skills required for synch. The critical disadvantage of synchronous design is its vulnerability in a high variability environment – a synchronous design requires that the latency of a logic path be constrained to avoid metastability, but this cannot be guaranteed under the variation experienced at subthreshold supply voltages. Instead this chapter proposed two architectures, asynchronous and a hybrid approach known as GALS as more suitable for subthreshold designs. Asynchronous designs offer higher reliability at the cost of additional design and verification effort, and GALS allows the incorporation of synchronous designs within an otherwise asynchronous design. Table 4.2 summarises the comparative methods of these approaches.

Although asynchronous design has been identified as suitable for subthreshold designs, different implementation styles exist, not all of which are suitable for subthreshold use. The fundamental issue that differentiates the competing asynchronous methodologies is the presence of timing assumptions and the ability to reduce (but never eliminate) the probability of their violation. Of the design styles proposed in this chapter (summarised in Table 4.3 on the next page), dual rail designs require the fewest assumptions and can be made arbitrarily robust at the cost of latency and area overhead.

Mixed timing and GALS designs offer additional challenges. Whenever two different timing domains (defined by implementation style or clock requirements) communicate between each other, there is a need to ensure that such communication is performed in a safe way and that timing hazards are avoided. The method proposed in this chapter helps to address this critical requirement by identifying the boundaries between timing domains so that the synchronisation techniques can be applied.

Timing interfaces for use in Mixed Timing Domains will be presented and evaluated at subthreshold in the next chapter. The benefits of asynchronous design at subthreshold will also be explored.
### Table 4.3: Summary of asynchronous design methodologies as applied to subthreshold design

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Description</th>
</tr>
</thead>
</table>
| Bundled Delay Line| Careful modelling needed to select critical path  
                        Closest to synchronous methodologies  
                        Vulnerable to process variation |
| Current Sensing   | No requirement for determining critical path  
                        Incorporates analogue components into digital design  
                        Difficult transition to different process technology  
                        Requires mixed-signal simulations  
                        Effect of process variation on current sensors has not been assessed. |
| Dual Rail         | Completion easy to determine  
                        Fewest timing assumptions  
                        Resistant to process variation  
                        More complex gates required  
                        Wider buses needed |

---

**Table 4.3:** Summary of asynchronous design methodologies as applied to subthreshold design
Chapter 5

Arbitrary Mixed Timing Domain Systems

Chapter 4 introduced the principle of Mixed Timing Domain systems (MTD). To realise an MTD system, the fundamental requirement is to be able to communicate across different timing domains through the use of timing interfaces. This chapter addresses this requirement by providing a family of robust FIFOs capable of acting as timing interfaces between the different kinds of timing domain: synchronous to synchronous (mixed clock, MC), asynchronous to synchronous (AS), synchronous to asynchronous (SA) and asynchronous to asynchronous (AA).

FIFOs are often represented as a sequence of stages, with each stage containing storage and control logic. One way to classify FIFOs is according to the proportion of local and global logic. Local logic is repeated within each stage whereas global logic belongs to the FIFO as a whole and is not replicated within each stage. An overview of a FIFO made entirely of local logic is shown in Figure 5.1. Any of the local blocks within the FIFO stage can be extracted to become global logic. State Control determines the FIFO position and whether a given stage is full or empty.

A mixed clock FIFO is one in which the put and get interfaces are synchronous domains controlled by different clocks. Mixed clock FIFOs are classified according to a broad taxonomy based upon the frequency and phase relationship between the two clocks [93, 99]. Various optimisations can be made if the two clocks are constrained to a given relationship. By contrast, this chapter concentrates on arbitrary clock domains where no assumptions or constraints are made on the behaviour of the two clocks.
5.1 FIFOs in Literature

This work was inspired by the FIFO designs by Chelcea and Nowick [17], previously discussed in Section 4.4.5 on page 81. In the Chelcea design, the Full/Empty Controllers and synchronisers are global with the remaining logic local. Several problems were found in these designs which are addressed in the new FIFOs presented in this chapter. Arbitrary clock frequencies are supported, although the bounds of acceptable clock frequencies are specified at design time. Synchronisation occurs through synchronisers, the length of which are dependent on both the sender’s and the receiver’s clock frequency.

The authors of [72] were also inspired by the designs in [17]. In their FIFO, all logic is local, including the synchronisers. Their asynchronous design is built from standard cells, rather than C elements [97] common in other designs (including [17], [79] and this chapter). They claim that their design supports all four timing domain combinations although no evidence or data is provided. Further analysis of this FIFO, exposing problems in the design can be found in Section 4.4.5 on page 83.

Recall* that Rahimian [79] used a single FIFO design, with synchronous or asynchronous wrappers on either side. All logic to the FIFO is global. A slightly different

---

*Section 4.4.5 on page 84
mixed clock FIFO based upon pausable clocking is presented in [80], with apparently identical performance. Properly implemented, pausable clocking would not suffer from the flaw discussed in the previous chapter.

Apperson [7] presents a mixed-clock only design where all logic is global. The all global approach has superior queue depth scaling, but this approach is uncommon for asynchronous FIFOs. Although the most area efficient synchronous FIFOs have memory based storage, this is rarely used in asynchronous designs. Specialised designs often provide superior performance or better area efficiency, however they are less portable as the interface is often unique to the FIFO. Families of FIFOs, although less efficient, allow for modular replacement with minimal changes to the FIFO users.

[101] presents a family of FIFOs for GALS applications which share similarities with both Apperson’s [7] and Rahimian’s [79] designs. All logic, other than data storage which uses a register approach, is global. The FIFO’s state is encoded using Johnson encoding, as opposed to a mixture of both binary and Grey encoding used in the Apperson FIFO. The FIFO is naturally mixed clock, so like [79] a wrapper is used to interface the design to asynchronous modules. Also presented is a pausable, variable frequency clock generator, which could easily be incorporated into a variation-aware subthreshold design.

### 5.1.1 Comparison to New FIFOs

The FIFOs proposed in this chapter are based upon the ones presented by Chelcea and Nowick [17]. The design philosophy was to impose minimal restrictions upon designers incorporating a FIFO into larger designs. Robustness and accounting for corner cases were the design goals. The primary difference between the presented designs and those in [17] is the way in which the FIFO is used. In the synchronous case, fewer timing restrictions are imposed upon the FIFO users. For the asynchronous senders and receivers, dual-rail logic was adopted for enhanced reliability.

State encoding in the new FIFOs is implemented using one-hot encoding, in contrast to [7, 101], as this is simpler to implement at small queue depths. Unlike [79, 101] who use an asynchronous wrapper to connect a synchronous FIFO to asynchronous domains, the new designs consist of a family of four FIFOs, capable of natively
5.2 Mixed Clock FIFO

transmitting information between the possible combinations of asynchronous and synchronous systems.

5.2 Mixed Clock FIFO

The first FIFO to be presented is the Mixed Clock (MC) FIFO. The design acts as an interface between two domains with arbitrary clock frequencies, where no restrictions are imposed on the nature of the clocks. In a GALS context, the MC FIFO would be used for point to point interconnect, connecting synchronous timing domains without an intermediate asynchronous layer.

Figure 5.2 shows the top level structure of the FIFO. A FIFO with three stages is displayed, but any length greater than one is supported. As in Chelcea, the mixed clock FIFO contains a mixture of local and global logic. The Full and Empty Detectors and the Put and Get Controllers are global. All other logic is local. The
architecture within each stage is shown in Figure 5.3. At this level of abstraction there are minor differences compared to [17], but the overall behaviour of the FIFO varies due to the different designs of the sub modules.

5.2.1 FIFO Operation

A put request (a timing diagram of which is shown in Figure 5.4) is initiated by raising put req and placing the data word on the put data bus. Whether a put operation is allowed to proceed is determined by the Put Controller, which implements Equation (5.1). Once a write operation has been requested, the Data Validity Controller changes state, and the FIFO stage will now be listed as no longer empty (E[i] falls). The actual writing of the data into the FIFO’s storage registers occurs at the clock edge subsequent to the request, at which point the put token, ptok, advances to the next stage. At this point the stage is formally recognised as full (F[i] is raised) and this status is passed to the Empty Detector, which will propagate this information through a synchroniser to the get domain. After the synchroniser delay (determined
by the period of $\text{clk\_get}$ has elapsed, $\text{empty}$ will fall, the change in state indicating there is data ready for access.

$$\text{en\_put} = \overline{\text{full}} \cdot \text{put\_req} \quad (5.1)$$

To receive data, system implements the timing sequence shown in Figure 5.5. The logic controlling the get side of the FIFO begins by raising the $\text{get\_req}$ signal. If $\text{empty}$ is high, no action will be taken until it falls and the receiver will not be required to keep $\text{get\_req}$ high during this time. If however $\text{empty}$ is low, the data will be placed onto the $\text{get\_data}$ bus and $\text{gtok}$, the token representing the FIFO’s get position, advances at the next rising clock edge. When the output bus is set to the FIFO stage’s data, the stage is reported as being no longer full ($F[i]$ falls),

---

**5.2 Mixed Clock FIFO**

---

**Figure 5.4:** Timing diagram for sending a word to the MC FIFO

**Figure 5.5:** Timing diagram for retrieving a word from the MC FIFO
however it is not reported as empty until the next rising clock edge. This behaviour is explained in more detail in Section 5.2.2.

The Full Detector stops the FIFO early to prevent overflow as the FIFO is full if there is at most one empty stage. The Full Detector includes a synchroniser, tied to the $\text{clk}\_\text{put}$ signal to ensure the output is free from metastability. Detection of an empty state is split into two signals, nearly empty and original empty, which are combined into a single empty signal according to Equation (5.2). Nearly empty ($\text{ne}$) is high when one or fewer stages are full whereas original empty ($\text{oe}$) is true when no stages are full. In each case, the output of the detector is passed to a synchroniser controlled by $\text{clk}\_\text{get}$ to ensure safe operation. Within the Original Empty Detector’s synchroniser, the signal is ORed with $\text{en}\_\text{get}$, as shown in Figure 5.6 (a). This temporarily forces original empty high for one cycle following a get operation, and is needed to briefly stall the FIFO when nearly empty is high to avoid underflow.
5.2 Mixed Clock FIFO

The Nearly Empty Detector is shown in Figure 5.6 (b). The architecture of the Full Detector is equivalent to that of the Nearly Empty Detector, except that it processes the \( e[i] \) signals from the FIFO stages and is clocked using \( \text{clk}_\text{put} \). The use of two signals to determine empty status is a deadlock-avoidance measure which has been retained from the Chelcea design.

\[
\text{empty} = \text{ne} \cdot \text{oe} \quad (5.2)
\]

In peak performance, one put operation and one get operation may occur per cycle. The success of put operations is implicit. A put operation will always succeed providing \( \text{full} \) is low, likewise a get operation is guaranteed to be successful when \( \text{empty} \) is low.

**Behavioural Differences with Chelcea [17]**

In Chelcea’s design, the write operation is triggered at the start of the clock cycle and the DVC processes the operation immediately. In contrast, my design triggers the write operation at the end of the cycle, allowing \( \text{put}_{\text{req}} \) to rise, or glitch, at any time before the rising put clock edge, providing setup time and hold time requirements are satisfied. No action takes place until the rising clock edge, where the data is written to the register, the put token is passed onto the next stage and the Data Validity Controller (DVC) changes the state to full.

Get operations behave similarly. Chelcea’s design requires \( \text{get}_{\text{req}} \) to be raised at the start of the cycle, at which point the DVC immediately marks the stage as empty and the data is placed on the bus. My design defers marking the stage as empty until the subsequent rising clock edge, while still placing the data on the bus immediately. Like my put implementation, no restrictions are made on the timing of the \( \text{get}_{\text{req}} \) signal, beyond ensuring setup and hold time constraints are met. It may stay high to receive up to one word of data per cycle, or it may fall at any time before the next get clock.

As in the put case, the transaction isn’t completed until the next rising get clock edge. If the receiver lowers \( \text{get}_{\text{req}} \) before the get clock rising edge, the data will remain in the FIFO until a future transaction. This allows tolerance for glitching, unlike the design in [17]. In my design, the success of a get operation is implicit.
in contrast to Chelcea’s [17] use of a valid signal, which is redundant but included for interface compatibility. This simplifies the design for the Get Controller, which implements Equations (5.2) and (5.3).

\[
\text{en_get} = \text{empty} \cdot \text{get_req} \quad (5.3)
\]

By removing the parallelism from the design my implementation increases latency, however this eliminates the need for glitch protection on the request signals and prevents FIFO overflow and underflow caused by variations in clock frequencies.

I also replaced the output tristates in Chelcea’s design with muxes. The desire for low power consumption precludes the use of a weak keeper and it is difficult to guarantee a tristate bus is correctly driven at initialisation without the use of a weak keeper.

### 5.2.2 Data Validity Controller

The Data Validity Controller, shown in Figure 5.7, crosses three timing domains: synchronous put, synchronous get, and a non-synchronous domain. Once this observation is made, it is possible to provide additional information to the synchronous portions of the chip, without increasing the probability of metastability. The state transition graph (STG) of the asynchronous section is shown in Figure 5.8.

STGs are a form of Petri-Net [21, 22] and are commonly used in asynchronous design. STGs have two components, vertices and arcs. Vertices, which are the named transitions in Figure 5.8, represent the input and output signals depicted in
5.2 Mixed Clock FIFO

![Figure 5.8: STG describing part of the non-synchronous portion of MC DVC](image)

Figure 5.7. Arcs indicate relations between transitions and hold the state of the system. These relations are causal; a transition may trigger only after all of its input transitions have triggered. The initial state of the STG is represented by filled circles. The STGs were synthesised using Petrify [22] with the .slowenv option. This imposes the constraint that the system must settle before the next input transition. This constraint is easily satisfied when incorporated into the FIFO under all but the most extreme intra-die process variation.

The logic in the STG is the key improvement to the existing design. Admittedly it is much larger than the original, but it permits more flexibility for the users of the FIFO. A simpler design is possible by waiting for the clocked signals to fall before triggering a change in output, however this will increase latency by one cycle.

<table>
<thead>
<tr>
<th>Full Signal (F[i])</th>
<th>Empty Signal (E[i])</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Neither</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Empty</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Full</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal State</td>
</tr>
</tbody>
</table>

Table 5.1: Output states for MC Data Validity Controller

The Data Validity Controller has three states: full, empty, and neither, as shown in Table 5.1. The logic characterised by the STG outputs a binary full/empty value (F). Neither is an intermediate state that allows for temporary state updates before a put or get operation has been completed and is obtained by additional processing. Whenever en_put or en_get are raised and a stage holds the relevant token, they temporarily force the respective empty or full signals to 0, invoking the neither
state. Although this reintroduces the possibility of glitching, this is not an issue. As the \texttt{enPtok} signal also belongs to the put domain, this can be freely invoked without increasing the chance of metastability assuming setup time conditions are met. Should it be triggered accidentally, it will revert to the correct state on the next cycle and at worst will briefly stall the FIFO. The get domain behaves similarly with the \texttt{full} and \texttt{enGtok} signals. The purpose of this feature is to propagate the change in state more quickly, compensating for the delay introduced by processing at the end of the cycle, rather than the beginning.

The new Data Validity Controller also resolves a significant flaw in the synchronous parts of the Chelcea DVC [17]. In their design, exceptionally tight timing is required for the \texttt{get.req} signal. \texttt{get.req} must stay high long enough so that the enable signal, \texttt{en.get}, satisfies the register’s hold time, yet must fall quickly enough to ensure that the \texttt{en.get} of the subsequent stage does not momentarily rise once the get token is passed as this will toggle their DVC’s SR flop. The synchronous put domain signals suffer from this flaw as well. Such tight timing constants cannot be tolerated if the FIFO is to be used as a timing interface for a Mixed Timing Domain system at subthreshold where variability would make them impractical to satisfy. The new design, however, is not affected by this problem due to the use of D-type flops on all users of \texttt{en.get}. So long as \texttt{get.req} stays high long enough to satisfy the hold time of the flops, it does not matter when it is lowered, providing the setup time constraint of the next cycle is not violated.

### 5.2.3 Full/Empty Detectors

The dynamic Full/Empty Detectors in [17] were also removed and replaced with static equivalents. This change was needed to satisfy our requirements for support of arbitrary, varying and stoppable clocks. Dynamic leakage in the [17] design can cause serious problems at slow clock rates as discharge of the storage nodes can cause the FIFO to falsely report not-empty/not-full and can cause high power consumption as both pull up and pull down networks of the load are active.

The conversion of the Full and Empty Detectors to a static design removes one aspect of the metastability protection claimed in [17], where metastability is only possible on one transition of \texttt{full}/\texttt{empty}. Our static design can be affected by metastability
5.2 Mixed Clock FIFO

Figure 5.9: MC FIFO high level design

on both rising and falling transitions, but allows operation over a much wider range of operating conditions.

The length of the synchroniser within the Full and Empty Detectors is determined solely by the relevant clock, unlike [17] where consideration of both clock frequencies is necessary. The new DVC and Full/Empty Detector design permits arbitrary, varying and pausable clocks. In doing so, no arbiter is required and each clock domain requires no knowledge of the other domain.
5.2.4 Timing Domain Analysis

Taking into account the domain assignments of the previous sections, Figure 5.9 applies the timing domain methodology to the overall design of the Mixed Clock FIFO, with the MC stage modules divided into stages per Figure 5.3 (Note that the coloured regions indicate which domains are present in a module, but not the geometry of the domains.) The timing interface synchronisers within the Full and Empty Detectors are represented as a black band.

What remains is to explain how the output data, \texttt{get\_data} is properly converted to the get domain. Consider \texttt{get\_data[2]}, the data output of the rightmost MC stage. Upon leaving the stage, the signal bus still belongs to the put domain. This is because the bus is connected to the output of a storage register belonging to that domain. The bus is then connected to a multiplexor, whose control signals (\texttt{gtok[0:2]}) belong to the get domain. As the implementation of the mux is purely combinational, it is possible to defer the placement of the timing interface until the output of the mux. To make the data bus safe for use in the get domain, the output is guarded by the \texttt{en\_get} signal through an AND gate\footnote{See Section 4.4.1 on page 64 for the definition of a guard signal}.

To use a guard signal it is necessary to guarantee that when active, the guarded signal will not change. When the put and get tokens point to different stages of the FIFO, the stage with the put token cannot be written to as \texttt{enPtok} remains low, so the data will be guaranteed to remain constant.

There are also two situations where the put and get tokens point to the same stage of the FIFO. In the first situation, the FIFO is empty. In this instance, the \texttt{empty} signal forces \texttt{en\_get} low until the put domain writes to the stage’s register, passes the put token to the next stage and updates the Data Validity Controller. After passing through the synchronisers, \texttt{empty} will fall, allowing the information to be read.

The second possibility occurs when the FIFO is full. Here, the full signal will prevent further data to be written until a read operation is performed. On the rising get clock edge, the DVC status will be reset to empty and the get token will be passed to the next stage. After this information is propagated through the put side synchroniser the sender can write to that stage without any adverse effect on the receiver.
5.3 Asynchronous to Asynchronous FIFO

5.2.5 Resistance to Timing Failures

In a mixed clock synchronous design there are two principle potentials for timing failure: metastability and intra-die process variation.

Resilience to metastability in our design is determined entirely by the strength of the synchronisers, of which the probability of failure can be evaluated using methodologies such as [111]. When communicating between modules with different supply voltages, the FIFO should be operated at the higher voltage. This is because the width of the window in which metastability can occur is exponentially dependent on supply voltage at subthreshold [61], so higher supply voltages will have a marked improvement on the strength of the synchroniser.

Process variation is a more serious concern. No synchronous design can be entirely immune from the effects of intra-die process variation due to the potential to cause clock skew. Instead the goal is to maximise yields, defined as the proportion of chips tolerant to intra-die process variation. By partitioning a design into multiple domains, which communicate using FIFOs such as the ones proposed, the effects of intra-die process variation can be isolated. Should a portion of the chip lie closer to the worst-speed corner, a slower clock could be generated for that module, leaving the rest of the chip unaffected.

5.3 Asynchronous to Asynchronous FIFO

The Asynchronous to Asynchronous (AA) FIFO is simpler than the other variants described in this chapter. Unlike the other designs, it only transmits data within an asynchronous timing domain, not between multiple domains. It is included to complete the FIFO family, to provide a base for the remaining designs and to act as a representative of asynchronous system. Converting between different asynchronous protocols can easily be accomplished through the use of format converters, as in [72].

The FIFO uses a speed insensitive dual rail design with four-phase handshaking in contrast to the single rail implementation used by Chelcea. This was chosen to enhance robustness by reducing the effect of layout induced timing variations and intra-die process variation.
5.3.1 Design

The FIFO architecture is shown in Figure 5.10. Unlike the FIFOs with synchronous components, all logic is local, although there is still a need to combine acknowledge and data signals across stages. The logic within each stage is shown in Figure 5.11. As a dual rail design, the interface logic is simplified as there is no need for put request and get acknowledge signals. Instead, this information is carried within the put_data and get_data buses. As a result of this change, the Obtain Get Token logic in [17] could be used for both the Put and Get token Controllers from Figure 5.11. The Data Validity Controller was also implemented unchanged from [17], the STG of which is shown in Figure 5.12. The OR gate connected to the get_data behaves as a mux as it is guaranteed that at most one FIFO stage will have non-zero output at any time as the register will keep the output low unless RR is raised.

Like the Mixed Clock FIFO, position is determined by passing a token. These tokens are transmitted through the read enable, re, and write acknowledge, wa, signals, rather than by dedicated tokens. Write select, ws, enables the register for write access if the stage is not already storing data.
5.3 Asynchronous to Asynchronous FIFO

A put operation is initiated by placing the data on the `put_data` bus. If one of the stages is empty, its `ws` signal will be high and the data is written to the register of that stage. The register will respond by raising the write acknowledge signal, `wa`. At this point the AA Data Validity Controller will set `full` to high and the stage cannot be written to again until a subsequent read operation is completed. This sequence is shown in Figure 5.13.

To conclude the write operation, the sender returns the `put_data` bus to low, at which point the register lowers the `wa` signal. Once `wa` falls, the write token is passed to the next stage, where the next `ws` signal will rise if the FIFO isn’t full.
A read operation is initiated by raising the `get_req` signal. The stage holding the get token will raise `re` if the `full` signal is high. This will initiate a read request on the register, causing the stored data to be placed on the `put_data` bus. Once `get_req` falls, `re` will also fall. The read token is passed to the subsequent stage, and the `full` status is cleared.

### 5.3.3 Resistance to Timing Failures

This FIFO is implemented as a quasi-delay-insensitive (QDI) design. As previously described in Section 4.4.4 on page 79, Martin [66] proved that timing correctness of such systems cannot be guaranteed as all but a small subset of designs require timing assumptions (Isochronic forks in the case of QDI systems). However a subsequent paper of his [66] provides the solution to this problem. Isochronic forks do not require that all branches must transition at the same time, rather that signals whose transitions are explicitly acknowledged must transition later than signals whose completion is unacknowledged or assumed. The solution is to add a delay to the confirmed signals, to ensure that they transition after the unacknowledged ones.
5.4 Hybrid Domains

Rather than guaranteeing that the system will never fail, this method will provide a confidence of correctness, in a similar way to the probability of failure in synchronisers. In the case of an IBM 65 nm process, a delay stage of 7 inverters was needed to obtain a 1 in 2-million failure rate at a subthreshold supply voltage [66].

This process of demonstrating robustness to a sufficient level of confidence is left to the implementer, who would be in a better position to determine the desired yield for a given process technology and supply voltage. The latency values presented in this chapter should then be considered as a lower bound. However, the effects on energy efficiency of these measures will be considered later.

5.4 Hybrid Domains

The final parts of the family are the interfaces between the Synchronous and Asynchronous domains. Both Synchronous to Asynchronous (SA) and the Asynchronous to Synchronous (AS) FIFOs are supported. These FIFOs are useful for interfaces between GALS domains. To allow two way communication, one of each type would be needed.

These FIFOs are designed as hybrids of the purely Synchronous and Asynchronous FIFOs described earlier. In the Asynchronous to Synchronous FIFO, the put side is unchanged from the AA FIFO, whereas the get side is identical to the MC FIFO. The Data Validity Controller, described below, is the only difference. The SA FIFO, shown in Figure 5.14, behaves likewise, although it also includes a Single to Dual Rail Converter for the output.

The DVC for the AS FIFO is shown in Figures 5.15 and 5.16 and the STG for the asynchronous core of the SA DVC is shown in Figure 5.17. They are designed for low latency and parallel operation. In both cases reading is permitted before the write signal falls. The AS DVC also allows the write signal to rise before a read is permitted whereas the SA DVC requires the read signal to fall first. This is due to the difference in when a write is classified as complete. A synchronous get side defines a read as completed when $\bar{TogG}$ rises, but the rise and fall of a four-phase handshake must occur when there is an asynchronous get.
Chapter 5  Arbitrary Mixed Timing Domain Systems

Figure 5.14: An SA FIFO stage

Figure 5.15: Data Validity Controller for AS FIFO

Figure 5.16: STG for the asynchronous portion of the AS FIFO
The Single to Dual Rail Converter implements the logic specified in Equation (5.4). One of the requirements of the output of the converter is that it is immune to glitching. This is accomplished by AND-ing the data bit with an enable signal, which is \( re \) from Figure 5.14. If \( re \) is low, the output of the converter is forced low, so that subsequent stages will be unaffected by a change in the register’s state. When \( re \), is high, which occurs only when a read operation affecting the cell holding the read token is in progress, the FIFO stage is listed as full and the put side of the FIFO is forbidden to write to that stage. This assumes that the delay in the C-element producing \( re \) has a latency below the sum of the synchroniser delay and one additional clock cycle, which will be true in all but the most extreme cases of intra-die variability. As a result, providing the registers used to store the data word do not glitch without a change in input data, the output of the converter will not glitch.

\[
\begin{align*}
Y[1] &= A \cdot re \\
Y[0] &= \overline{A} \cdot re
\end{align*}
\]

(5.4)

5.5 Results

The FIFOs were fabricated in an AMS 3.3 V 0.35 µm technology, with a FO4 of 577 ps. The STGs used in the Data Validity Controllers were synthesised using Petrify [22]. A micrograph of a fabricated chip implementing an earlier iteration of the FIFOs is shown in Figure 5.18.
As maximising the operational range of the FIFOs was also a design goal, the fan-in of the gates used in the implementation was limited to 3. Although it was found in Section 3.2.2 on page 34 that high valency gates exhibit superior power consumption, it also found that the minimum voltage required for successful operation was increased. In this instance, operating range was considered more important.

Simulations were performed in Cadence Spectre at a temperature of 27°C for a typical corner. The FIFOs were simulated with 8 stages, each with a capacity of 4 bits.

5.5.1 Latency

Latency, excluding synchroniser delay for the AS and MC FIFOs, is plotted in Figure 5.19. Between 0.3 V and 0.7 V, an exponential improvement is observed, with latency improving at a slower pace beyond 0.7 V. When viewed over logarithmic scales, there is practically no difference between latency. This matches the behaviour of a NAND gate found in Section 2.2.3 on page 11, with the difference in the bounds of the exponential region being attributed to the different processes used. The NAND gate simulation used a 180 nm process while the FIFOs were constructed in a 0.35 μm process.
As the scale makes it difficult to compare the various configurations, the right graph of Figure 5.19 normalises the results with respect to the mixed clock FIFO. Although at first sight the FIFOs with synchronous receivers (MC and AS) have significantly better latency than the FIFOs with asynchronous receivers (AA and SA), this is due to a difference in defining latency. For the MC and AS FIFOs, latency is defined by the maximum duration between the receiver’s rising clock signal and the requested data being placed upon the bus, assuming that the data has already been stored within the FIFO. This method of defining latency is of interest to synchronous designers as it is needed for the determining of clock budgets and pipeline allocation. It is however inapplicable to FIFOs with asynchronous receivers, as they do not have clock timing budgets to satisfy. Instead, latency is defined as the duration between the put operation being enacted (the rising clock edge for the SA FIFO) and the completion of a get operation. As this incorporates additional work, a higher latency
Figure 5.20: Latency for the various FIFOs including synchroniser delay, normalised to the MC FIFO in the right graph

is to be expected. This approach measures the delay in the propagation of the data from the sender to the receiver.

To model latency of the MC and AS FIFOs in terms of data propagation, the figures for latency should include a synchroniser delay. This has been excluded in the results up to this point to remove the dependency on clock frequency. To demonstrate the effects, Figure 5.20 repeats the results with a synchroniser delay included. To model additional logic on the critical path of a system, the clock period chosen was five times the inverse of the MC throughput measured in Section 5.5.2 (a single clock frequency cannot be used given the exponential scaling of performance). This choice of clock frequency is arbitrary, but it demonstrates that including synchroniser delay results in the asynchronous receiver FIFOs showing the lowest latency when considering data propagation. This additional latency cannot be avoided without modifications to support clock pausing.
Figure 5.21: Left: Throughput for a combined put/get operation. Right: Normalised throughput

Figure 5.19 also highlights an issue with bundled delay in an environment where voltage scaling is performed. The relative latencies of the AS and MC FIFOs vary with respect to supply voltage, suggesting that the voltage scaling of the underlying gates is not uniform. To determine the required length of a delay line, allowances for variations in supply voltage must be made in addition to process variability. This is less of a concern for clock distribution trees given the limited variety of gates utilised, but verification of this assumption is necessary in order to validate a synchronous design for operation at widely varying supply voltages.

5.5.2 Throughput

Throughput, shown in Figure 5.21, also demonstrates exponential behaviour in this range of supply voltages. Throughput is calculated analytically, based on the inverse of combined rising and falling latency of the critical paths of sender and receiver. The
synchronous design of the MC FIFO lacks a falling latency component; throughput is instead determined by the inverse of the duration between the rising clock edge and the change in output state.

Throughput is harmed significantly by the need for acknowledgement of work completed in the FIFOs with asynchronous components. Upon placing a word in the FIFO, each register must acknowledge the bit has been written and these bits are combined through a completion tree into a single acknowledge bit. Further reductions in throughput are caused by the need of a given FIFO stage to verify that the subsequent FIFO stage has acknowledged the passage of the position determining tokens. The presence of the falling latency component in the calculation of throughput is caused by the use of four phase handshaking, as an operation is not considered completed before the input and acknowledgment signals have returned to a 0 state.

Through the use of a two phase protocol, such as LEDR [25], the need for a return to 0 at the completion of an operation would be removed, providing a more competitive throughput. This action however, will increase the circuit complexity as operations would begin on both rising and falling transitions. In practice though, throughput is not a problem as the FIFO will rarely be operating at peak performance.

### 5.5.3 Dynamic Power

Peak power consumption, including both static and dynamic components, is plotted in Figure 5.22 on the next page, calculated by measuring the average power consumption when the FIFOs were operated at their maximum rate of throughput. Over the simulated region, power consumption increases exponentially, but part of this increase can be attributed to the exponential increase in throughput, as the time interval over which the work is performed also decreases exponentially.

### 5.5.4 Leakage Power

Although the large feature size of the 0.35 µm process technology allows leakage power to be ignored at higher supply voltages, it is no longer negligible in the subthreshold and near-threshold regions. This is illustrated in Figure 5.23. The AA FIFO, when running at 0.3 V, had 97% power consumption attributed to leakage
5.5 Results

Figure 5.22: Supply vs maximum sustained power consumption

Figure 5.23: Percentage of energy attributed to leakage over a combined put/get operation
over a single put/get operation, compared to 0.03% at 1.0 V. Due to a higher switching activity, the MC FIFO has a lower rate of 93% leakage at 0.3 V. For the 0.35 µm process, leakage only becomes negligible when the supply is increased beyond the near-threshold region. The use of a smaller process technology would delay or eliminate the point at which the proportion of leakage power becomes negligible. In this case measures to decrease the impact of leakage power, for example through longer length gates[11] becomes increasingly important.

Leakage power for the FIFOs is plotted in Figure 5.24. Compared to the exponential scaling of peak power consumption*, the rate of increase in leakage power is much shallower. Leakage for the asynchronous FIFOs could be reduced if a single rail architecture was adopted. This would be especially noticeable in the deep subthreshold region, however additional techniques to mitigate intra-die process variability would be needed.

For the AA case, the leakage power can be equated with idle power. This can also be done with the other FIFOs with the assumption of perfect clock gating as idle clocks were excluded to remove the dependence on clock frequency. Figure 5.25 provides an

*Section 5.5.3
estimate of the effect that clock gating would have on power consumption. The clock frequency is set to the maximum rate achievable, assuming that nothing else is on the critical path. From this, it is clear that clock gating would have a significant effect on idle power consumption in the near-threshold region, but it would be unnecessary when operating at a deep-subthreshold supply voltage.

5.5.5 **Minimum Energy Operation**

In the subthreshold or near threshold regimes, raw performance is of secondary importance. The goal when operating at low voltages is to minimise energy used. While minimising power consumption is possible, the additional time required to complete an operation may increase the total energy used. Energy per operation is the new key metric*.

Energy per operation represents the combined energy for a single put and a single get operation. For the MC case, the simulation assumes that the FIFO is half full, and that the empty signal remains low. The dynamic energy consumed by the falling clock transition was included for the synchronous cases. The inclusion

*See Section 3.1.1 on page 30 for the definition
of synchronous overhead, such as synchroniser delay and unused clock cycles will further increase the difference between the two FIFOs.

Figure 5.26 shows the energy per operation as supply voltage was varied. To account for the effect of the significant contribution of leakage power, the interval used for the measurement of power consumption was determined by the worst case latency rather than the end of the testbench clock cycle. From the graph, at 0.3 V the MC FIFO consumes less energy per operation than the AA and SA FIFOs as it has the smallest leakage; however as $V_{DD}$ increases, the other FIFOs quickly prove superior. In a system with large numbers of registers, clock gating is imperative. At 1.0 V, approximately 80% of power consumption is caused by clock transitions.

The curve is also steeper for the mixed clock case. This can be attributed to a greater switching activity ratio. The minimum energy point for the two circuits differs by 0.05 V, however in this case the difference is not significant enough to warrant multiple supplies.

The overall shape of the curve and the different locations of minimum energy points has been reported for other subthreshold systems [2, 42, 107, 116]. The location of the minimum energy point also depends heavily upon the process technology.
5.6 Application to Subthreshold Design

Processes with smaller feature sizes have their minimum energy point at a lower voltage.

5.6 Application to Subthreshold Design

The previous section concentrated simply on evaluating the FIFOs. Here the observations and conclusions drawn will be extended to subthreshold systems more generally.

In addition to their role as timing interfaces, these FIFOs can also be generalised to represent interconnect between arbitrary modules which do not necessarily cross timing domains. The FIFOs could also be used as case studies to examine the behaviour of complex systems in the subthreshold and near-threshold domains.

With the synchronisers in the Full and Empty Detectors omitted and the two clock signals merged, the MC FIFO is representative of an ordinary single clock domain system, however as a single clock FIFO, it is not an optimal solution given the use of register storage. With different clocks, the MC FIFO models the point to point interconnect described in Figure 4.3 on page 62, or a synchronous interconnect framework.

The AA FIFO is representative of a dual rail asynchronous system as it does not cross timing domains. The AS and SA FIFOs model the interface between GALS synchronous modules and asynchronous modules or interconnect.

5.6.1 Application to GALS

From a power perspective, asynchronous interconnect is cheap when the power budget is not dominated by leakage. The FIFOs with asynchronous senders exhibited the lowest energy per operation. The ability to switch only when a transaction is to be performed has a significant effect on energy per operation and idle power consumption. The latency of such asynchronous systems is not significantly larger, and the use of 2 phase design or improved completion detection would mitigate most of the throughput penalty. The low cost of the asynchronous FIFO suggests that the interconnect framework of the design would be more efficient if it was asynchronous, leading to a GALS design.
In addition to reduced power consumption, GALS provides other benefits. In particular, vulnerability to intra-die variation is reduced, although it can never be entirely eliminated. GALS works by partitioning the design into smaller, independent timing domains. If one of these domains is afflicted by intra-die variation, reducing its maximum clock frequency, the clock can be reduced for that domain independent of the rest of the chip. In a design with a single clock domain, the entire system would need to run at the speed of the slowest component and the number of samples that fail to meet requirements would increase. In this way, a GALS design methodology could increase yields in subthreshold circuits suffering from severe process variation.

The comparatively high cost of the MC FIFO is a disadvantage for a generalised synchronous interconnect framework. However when used as point to point interconnect the MC FIFO is still superior to a combination of an SA and an AS FIFO.

More broadly, it was found in Section 5.5.5 that the voltage required for minimum energy operation depends on the nature of a logic module. To truly minimise power consumption, different supply voltages would need to be provided to the various components of a design. However a difference in minimum energy voltage levels greater than that shown in Figure 5.26 may need to be demonstrated for the improvements to be significant. As supply voltage has an exponential effect on transistor delay\(^\ast\), a single timing domain would be impractical in a multiple supply voltage design. Instead, a Mixed Timing Domain design would allow each module to operate at its own pace.

## 5.6.2 Application to Asynchronous Design

After taking the step to incorporate asynchronous design methodologies and letting components work at their own pace, further benefits can be obtained by adopting a purely asynchronous design.

Self timed quasi-delay-insensitive asynchronous designs are ideal for subthreshold systems. The high variability reported in Section 2.2.6 on page 19 can be greatly mitigated by letting the circuit work at its own pace. Intra-die variation is a particular problem for conventional designs, causing the critical paths to vary between chips.

\(^\ast\)See Section 2.2.3 on page 11
5.7 Further Work

This does not cause reliability issues for quasi-delay-insensitive systems, where only the isochronic forks are vulnerable to failure. The location of these forks are known at design-time and can be made arbitrarily robust through the addition of inverter chains [66].

Bundled delay line methodologies are not recommended for subthreshold operation, as the potential for timing failures return if process variation causes the delay line latency to be less than that of the linked logic. Similar to isochronic forks, bundled delay lines can be made sufficiently robust with a long enough inverter chain, although the size of the chain required quickly becomes impractical.

Although the AA FIFO offered the lowest leakage power of the four designs, it could be reduced further. As a dual rail design there is additional area overhead compared to single rail systems. Registers and completion chains are twice the size of single rail designs and gates to perform logic operations are typically more complex. For example, a 2 input NOR gate can be implemented using 4 transistors for single rail systems, but it requires 14 transistors for a for a dual rail design. (Inverters are a notable exception. They can be implemented without any transistors, simply by swapping the rail labels). This overhead translates to increased leakage power given the larger number of transistors.

If third party IP is involved, an all asynchronous design may not always be practical, so the partial approach of a GALS methodology, remains a suitable compromise.

5.7 Further Work

There are several modifications that could be made to improve the FIFOs. As performance was not a priority for the design of low powered FIFOs tolerant to process variation, no optimisations have been performed to improve latency or throughput. With appropriate care, these metrics can likely be improved without compromising the reliability of the FIFOs.

To improve power efficiency at higher supply voltages, clock gating could be implemented. Energy per operation and idle power would be significantly reduced if clock gating was extensively applied to the register array and token registers. This could be accomplished by using the token positions to determine which D-flops are gated.
Best case results of this are shown in Figure 5.25, but this may be overly optimistic. The use of clock gating would be especially noticeable in the MC and SA FIFOs, given the large register banks.

Section 5.6.1 proposed and qualitatively described GALS topologies. Quantitative measurements would be needed before firm conclusions could be drawn. Alternatively, an existing GALS system could be examined. While it would not use the new FIFOs, it would have the advantage of evaluating a complete system. Many digital designs will scale down to subthreshold or near-threshold supply voltages if conservatively designed.

5.8 Conclusion

The FIFOs examined in this chapter demonstrate that asynchronous designs can achieve low energy per operation, assuming these FIFOs are representative of larger systems. Low power synchronous systems are also possible, although with more caveats. At higher supply voltages, it is imperative that activity ratios are kept as low as possible through extensive clock gating. It is less important in the deep subthreshold region, where leakage dominates.

GALS and asynchronous design are well suited for use at subthreshold, due to their ability to lower power consumption and reduce the effect of process variation. However if designing for performance, results need to be compared at the target supply voltage as the relative merits of competing designs can vary.
Chapter 6

A Wakeup Timer Based on Starved Transistors

Previous chapters have achieved low power consumption by reducing the supply voltage to below the threshold voltage, however not every system is suitable for running at a subthreshold voltage. For these instances, a different approach will now be undertaken, based on setting the gate voltage of transistors permanently below the threshold voltage, a process referred to as starving the transistors.

To provide a context for the starved transistors, they will be examined as part of a proposed application, a wakeup timer. Placing a system in a powered down state is a common method to achieve energy efficiency. In order for the system to still perform useful functions at an appropriate time, a wakeup timer is required to restore the system into an active state. As a wakeup timer must remain active at all times, designing it for energy efficiency is critical for the system to meet power requirements.

After explaining the target application, this chapter will explore the properties of starved transistors. These transistors will then be incorporated into an oscillator for use in wakeup timers.

6.1 Slow, Low-Power Clocks

Clock generation is a mature technology. Although research is still undertaken to design high-frequency or high-accuracy clocks, little attention is paid to the rest of the design space.

The quartz oscillator, a common low cost clock generator has operational frequencies ranging from approximately 1 kHz to 100s of MHz [87, page 993]. To generate
6.1 Slow, Low-Power Clocks

slower frequencies, a counter is used to trigger the output once the desired number of cycles has elapsed. A commonly used frequency is 32.768 kHz, chosen for the ease of detecting when one second has elapsed.

Although any oscillator with a period less than the desired interval would suffice by using a counter set to trigger when an interval’s worth of cycles have elapsed, the system could be more power efficient by using a slower clock, reducing not only the size of the counter, but also the update frequency. To explain the benefits of such a low frequency, recall from Equation (1.3) on page 1 the causes of dynamic power consumption in a circuit. Ultra-slow clocks achieve low power operation by significant reductions in operating frequency, $f$. This is possible when there are long periods before the activity is required.

For slower clock generation, an operational amplifier based approach might be used. These are useful for frequencies between 10 Hz and 100 kHz, with the lower bound limited by the size of the required passive components [87, page 987]. As analogue devices, the power consumption of operational amplifier based clocks may be too large to be practical for low-power wake-up circuits.

Through the use of a 555 timer based circuit, frequencies of 10 mHz–100 kHz are obtainable. CMOS implementations, such as the LMC555[40] are capable of sub-milliwatt power consumption and are certified to operate at lower voltages than BJT based designs. Like operational amplifier based clocks, 555 timers require increasingly large passive components for low frequencies.

6.1.1 Slow, Low-Power Clocks in Literature

Descriptions of the circuits used to construct discrete oscillators can be found in [87, Chapter 12]. CMOS oscillator generation techniques are covered in [83, Chapter 14]. For most implementations these designs would be sufficient, but for low-power wake-up timers either the power consumption is too large or bulky discrete components must be used. To address these problems, several oscillators have been proposed.

[62] proposed a wakeup timer based on the charging and discharging of capacitors built out of MOSFETs and relying on gate leakage to function. The clock supports supply voltages between 0.3 V and 0.65 V and can achieve an output period of
between 10 s and 17 s, depending on supply voltage and temperature. Power consumption ranges from 100 fW to 1 nW over the range of supported supply voltages.

To generate a clock signal to control a charge pump capable of generating negative voltages with low power dissipation, [60] presented a five stage ring oscillator, with each stage constructed out of starved transistors. Whereas [62] relied on gate leakage, this approach utilises subthreshold channel leakage. The oscillator operates in the near-threshold region of 0.6 V, oscillates at an average frequency of 4.6 Hz and consumes 0.64 pW. This chapter describes a similar approach, which I have previously published in [51]. Comparisons between this design and that of [60] are performed in Section 6.4.2.

[63] also proposes a wakeup timer, which is built from an SR flop, controlled by an input bias voltage. A frequency of 11 Hz and a power consumption of 100-150 pW at a supply voltage of 0.6 V is reported. Power consumption and frequency are principally determined by the bias voltage and the rate at which it is refreshed. In comparison to [62], reduced supply sensitivity is reported.

To address the problem of temperature variation present in the other designs, [59] proposes a variable stage design, with the number of stages determined by a reading from a temperature sensor. With this compensation, the variability is reduced by a factor of 15 and 51 compared to [63] and [62] respectively. A higher power consumption of 660 pW was reported. The supply voltage at which this was measured was not mentioned although the timer supports 650 mV to 1.25 V operation.

With the exception of [59], a common disadvantage of ultra-slow clocks is the markedly reduced accuracy compared to conventional timers. [62] observed intra-die frequency variations of roughly 20% and inter-die variation of approximately 100% from a sample of 25 dies. [60] found near-exponential temperature induced variability. This is often a consequence of the target supply voltage, which has significant variation in the delay of conventional logic as has been shown in the subthreshold region in Section 2.2.6 on page 19.
6.2 Target Application

As the literature review demonstrated, the principal application of an ultra-slow clock is a wakeup timer, for systems when there is a long period between wakeup intervals (for example one second). Systems with highly constrained energy budgets are especially suitable, given the reduction in power consumption associated with an ultra-slow clock.

Although more general uses are possible, the wakeup clock presented in this chapter was initially created for a particular application, an electronic smart label. Although the label never eventuated, it is described below to provide the context in which the wakeup timer was designed and to provide justification for the design decisions made.

6.2.1 Electronic Smart Label Project

With the continued decrease in the cost and size of electronics, there are opportunities for its integration into new products. A form of electronic label, the RFID tag [108], is widely adopted in industry, with product identification and theft protection being the principal advantages. These goals are important for manufacturers, distributors and retailers who are involved in the transportation and sale of the goods, but provide no direct benefit to the consumer.

The electronic smart label project [77] sought to address this by creating a label which provided additional value to the consumer directly. Unlike an RFID tag, which requires a reader to extract the embedded data, the means to report information would be included in the label itself.

Figure 6.1 depicts a conceptual diagram of the proposed prototype, which would be affixed to wine bottles. Whereas an RFID tag draws power from and reports data using an antenna to transmit short range radio frequencies, the prototype stores power within a battery and uses a loudspeaker to communicate.

One of the features of the proposed smart label was an ability to log temperatures and report whether a specified temperature threshold has been crossed. As this is an indication that the wine has not been stored appropriately, this provides a way for the customer to assess the suitability of wine before purchase. The temperature logging behaviour is described in Figure 6.2. Although most of the label is powered
Figure 6.1: A conceptual diagram showing the physical construction of the smart label (from [77])

Figure 6.2: A flow diagram showing the temperature logging behaviour of the smart label (from [77])
down during sleep mode to preserve battery life, the wakeup timer must remain active at all times so that the label will wake for the next temperature reading. To accomplish that a slow clock is used, which will be described further in this chapter.

### 6.2.2 Requirements

The specifications of for the smart label allowed a significant amount of flexibility regarding the design of the wakeup clock. Other than stating that clock was slow (i.e. a long period), the specifications for the smart label [77] did not provide details regarding accuracy or period between wakeup events. The estimated power consumption for the clock was 1 µW, although minimising actual power consumption was desirable for its effect on battery life. The VLSI chip, which would contain the wakeup timer, was specified to accept a supply voltage from 0.8 V to 1.5 V, depending on the state of the battery.

### 6.2.3 Other Applications

Wakeup timers are not the only application for ultra slow clocks. Watchdog timers are a common feature in electronic devices to ensure the system is functioning appropriately. If the device does not complete a specific action within a specified period governed by the watchdog timer, the system is considered to have failed and is reset to a functional state. The exact time taken to determine system failure is not important, however it is desirable that false positives are minimised as data and state loss will occur when the system is reset.

User interface timers are another area where slow clocks could be used. Compared to the speed at which even subthreshold electronics operate, human interaction with the system is slow. An ultra slow clock could be used to indicate when to poll the interface for new input. Alternatively, it could be used as a debounce mechanism. When a switch is pressed the output will repeatedly transition between active and inactive before settling down into the correct state. A debounce timer disables the switch for a period of time, preventing the spurious activity from affecting the rest of the system.
Each of these examples provide an appropriate instance for the incorporation of a clock into an otherwise asynchronous design, as promoted in previous chapters. Clocks can be fashioned from asynchronous logic through the use of a delay line and a counter, but for long period clocks a long delay line is impractical and a large counter suffers from the same power inefficiency as that for a fast clock source.

### 6.3 Starved Gates

One method to achieve a slow clock is to use transistors with slow transitions. For this, starved transistors can be used. A transistor is referred to as starved if the gate is permanently connected to a voltage level such that $V_{GS}$ is lower than the threshold voltage. The starved transistors in Figure 6.3 (a) are indicated by an asterisk, and all use a gate terminal connected to the transistor’s source rail. In this work, gates are called **fully starved** if both the pull-up and the pull-down networks contain starved transistors. A gate is **semi-starved** if only one of the networks contains starved transistors. An inverter with only the pull down network starved
6.3 Starved Gates

is depicted in Figure 6.3 (b). Collectively, the two concepts will be referred to as starved.

The distinguishing feature of starved transistors is their large width, compared to the other transistors. Note that despite the pull up network lacking a starved transistor, it still contains a very wide MOSFET. The importance of this feature will be explained in Section 6.6.2.

6.3.1 How Starved Gates Work

The conventional approach to abstracting transistors as switches in digital circuits is not applicable to starved gates. By modelling the starved transistors as open circuits, there is no way for information to propagate to the next stage.

The solution to this problem is the realisation that transistors do not behave as perfect switches, as there is always a residual current flow, known as subthreshold conduction. The equation for this, as specified by [110, page 88], is provided in Equation (6.1), assuming that $V_{DS}$ is sufficiently greater than $v_T$. In these equations $V_t$ is the threshold voltage, $v_T$ is the thermal voltage, $n$, $\mu$ and $C_{ox}$ are process dependent terms.

\[
I_D = I_0 e^{\frac{V_{GS}-V_t}{n v_T}}
\]
\[
I_0 = \beta v_T^2 e^{1.8}
\]
\[
\beta = \mu C_{ox} \frac{W}{L}
\]

Given the existence of this current flow, how can the output state of the gate be determined? The answer relies on the fact that the drain current of the pull-up and pull-down network are not necessarily equal. This is not to say that Kirchoff’s Current Law doesn’t hold, rather that the gate and body currents are non-negligible.

Consider the case of the fully starved inverter depicted in Figure 6.4. To determine the output voltage $V_Y$, it is necessary to find the voltages at points X and Z ($V_X$ and $V_Z$). If the input to the gate is high, the non-starved transistor in the pull down network is active. As a result, the voltage at the source of the starved nMOS transistor, $V_X$, is 0 V. When this occurs, the gate-source voltage of the starved nMOS
transistor, $V_{GS}$ is equal to 0 V, as both the gate and the source have a voltage level of 0 V.

The value of $V_Z$ is harder to determine, as it is floating because the pull-up network is turned off. The solution to this is to model the transistors operating in the subthreshold region as resistors, with current determined by Equation (6.1). As transistor $N1$ is operating in the saturation region, it is modelled as an open circuit since the resistance of a transistor operating in the saturation region is negligible compared to the resistance when operating in the subthreshold region. The first step is to determine the ratio of the current flowing through each resistor. The currents of each resistor considered individually are shown in Equation (6.2) where $r_w$ is the ratio between the widths of the starved and non-starved resistors and $p_1$, $p_2$, $n_1$ and $n_2$ are process dependent constants.
6.3 Starved Gates

\[ I_{P1} \propto p_1 e^{\frac{V_i}{p_1}} \]
\[ I_{P2} \propto p_1 r_w e^{-\frac{V_{GSP2} + V_i}{p_2}} \]
\[ I_{N2} \propto n_1 r_w e^{\frac{V_i}{n_2}} \]  

(6.2)

Here \( V_{GSP2} \) is an unknown as \( V_Z \) is unknown, but assume for the moment it is at the minimum possible value of 0 V, as the gate of transistor P2 is tied to \( V_{DD} \). This would also make \( V_Z \) equal to \( V_{DD} \) and provide the greatest current flow. The resistance of the transistors can be approximated by the inverse of the current. By Ohm’s law, \( R = \frac{V}{I} \). \( V \) is capped by the difference between the supply and ground rails which is typically in the range of 0.6 V to 3.3 V. By contrast the drain current of a transistor operating in the subthreshold region is significantly less than 1 \( \mu \)A, possibly in the order of attoamps. As the inverse of this is significantly greater than the magnitude of the voltage, the resulting resistance can be approximated by the inverse of the drain current.

As a result, the value of \( V_Z \) can be modelled with a voltage divider. As \( I_{N2} \) is less than \( I_{P2} \) and both are less than \( I_{P1} \), the voltage \( V_Z \) will tend away from \( V_{DD} \). This will have the effect of increasing \( V_{GSP2} \) and reduce \( I_{P2} \) until an equilibrium point is reached between \( I_{P1} \) and the combined effects of \( I_{P2} \) and \( I_{N2} \).

The next step is to determine the output voltage, \( V_Y \), which can be determined using the same voltage divider principle. With the reduced value of \( V_Z \) and hence increased value of \( V_{GSP2} \), the current flowing through the pull-up network of \( R_{P1} \) and \( R_{P2} \) is exponentially smaller than the current flowing through the pull-down network. As a result, \( V_Y \) will tend towards a low value. It may not reach the voltage of the ground rail, but it should be small enough that a logical 0 can be propagated to the next stage in a logic chain. If this is not the case, the widths of the pull-down network can be increased.

The preceding steps can also be used to model the behaviour of the gate with input low, assuming \( p_1 \) and \( p_2 \) are sufficiently close to \( n_1 \) and \( n_2 \) to make width the determining factor in the voltage dividers and that pull-up and pull-down transistor widths are symmetric to ease calculation. Should the former be an incorrect assumption, the widths of individual transistors can be manipulated to make up for this discrepancy.
Suppose now that the input switches from a high voltage to a low voltage. For ease of modelling, this will be assumed to happen instantaneously, although in a real circuit this will occur over a non-zero time interval. When this occurs transistor $P1$ will start to operate in the saturation region and transistor $N1$ will operate in the subthreshold region. At the moment of transition, $V_X$ has a voltage equal to 0 V and $V_Z$ will quickly rise to a value equal to the supply rail now that $P1$ is active.

Like the previous stage, the task is to find the new steady state value of $V_Y$ by first finding the new values of $V_X$ and $V_Z$. The currents flowing through each transistor considered individually is shown in Equation (6.3).

\[
I_{P2} \propto p_1 r_w e^{-\frac{V_X}{p_2}} \\
I_{N1} \propto n_1 e^{\frac{V_Y}{n_2}} \\
I_{N2} \propto n_1 r_w e^{\frac{V_{GSN2}-V_Y}{n_2}}
\]

(6.3)

At the point of transition $V_{GSN2}$ is equal to 0 V, as $V_X$ is 0 V. What happens next is determined by the voltage divider consisting of a combination of $P2$ and $N2$ for the first term and $N1$ for the second term. The lower resistance caused by increased width of $P2$ and $N2$ will result in $V_X$ trending away from 0 V. This will then cause a reduction in $V_{GSN2}$, resulting in a significant reduction in $I_{N2}$ until an equilibrium is reached.

These changes will have a corresponding effect on voltage $V_Y$. Treated as a voltage divider, the pull up network resistance consisting of $R_{P2}$ will be significantly lower than that of the pull down network, assuming the values of $n_1$ and $n_2$ are sufficiently similar to that of $p_1$ and $p_2$. As a result, $V_Y$ will trend towards the supply rail, although it may not reach it before an equilibrium is reached. However it should be large enough to propagate a logical 1. If this is not the case, increasing the width of transistor $P2$ will improve the maximum achievable voltage.

To verify the hypothesis that the currents of the pull-up and pull-down networks are not equal and that the output voltage, $V_Y$, sways to whichever of the networks has the greatest current flow, the currents at points $X$ and $Z$ were measured with a high and low input voltage. The fully starved transistor configuration from Figure 6.3 was simulated using a 0.35 µm process and a supply voltage of 1.5 V. The result of this experiment is shown in Table 6.1. Where the pull-up or pull-down network is
6.3 Starved Gates

<table>
<thead>
<tr>
<th>Input Voltage</th>
<th>$I_X$ (Pull-down)</th>
<th>$I_Z$ (Pull-up)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 V ($V_{DD}$)</td>
<td>-485 fA</td>
<td>-46.2 aA</td>
</tr>
<tr>
<td>0 V</td>
<td>52.7 aA</td>
<td>4.17 fA</td>
</tr>
</tbody>
</table>

Table 6.1: Steady-state drain currents of the starved transistors in a fully-starved gate

active in a conventional circuit, the current across its starved transistor is measured in femtoamps. Where the network would be inactive, the current is instead measured in attoamps. These results match the expected behaviour as the exponential dependence on $V_{GS}$ and the additional impedance of the narrow inactive transistor produce marked differences in the resulting current.

6.3.2 Characterising Starved Gates

The characterisation of starved gates will not be as rigorous as that in Chapter 2 for subthreshold gates due to the difference in purpose. Subthreshold gates are intended for general purpose logic, capable of performing arbitrary computations. In contrast, the purpose of the gates presented in this section is to act as a slow switch for clock generation purposes. It is therefore sufficient to limit the analysis to considering an inverter with fanout 1.

Although, there are many possible configurations of starved gates, two will be presented here to provide an introduction to starved gates. Additional gates will be studied in Section 6.6 as applied to a ring oscillator.

Methodology

A chain of six identical gates was simulated, with measurements performed on the third element. It is critical to test the gates with an appropriately shaped input as it will be shown that the achievable voltages of a given gate will have an impact on that obtainable by subsequent gates.

The definition of propagation delay used in Chapter 2 is of limited value when analysing starved gates. In that chapter, propagation delay was defined as the duration between the input and the output signal crossing the midpoint voltage level.
When there is a large discrepancy between the rise and fall times, it is possible for the output to transition faster than the input, leading to a negative propagation delay. This does not mean the system is non-causal, instead it indicates a susceptibility to noise. However, because the rise and fall times are still much slower compared to an ordinary gate, the danger that noise poses is somewhat mitigated.

The solution to this problem is to measure the propagation delay of a buffer comprised of the third and fourth gates. Average propagation delay can then be obtained by halving the measured delay. Although this method cannot provide figures for worst case propagation delay, it is not as important for starved gates as use in general-purpose logic is not recommended.

Circuits were simulated using an 0.35 µm process. Typical corners, a temperature of 30°C Celcius and a supply voltage of 1.2 V were used, unless otherwise mentioned. The temperature was chosen as a compromise between average room temperature (approximately 23°C) and core body temperature (37°C), environments in which the circuits are likely to operate. Although electronics are typically tested at higher temperatures, this would not be appropriate for the proposed oscillators. The power consumption of the wakeup timers is low enough to have negligible heating effects and the presence of other heat producing circuits can be discounted by the long intervals where the components are powered down. Cadence Spectre was used to simulate the designs using the Spectre libraries for the AMS 0.35 µm process. Other tools, such as Synopsys HSpice, can generate different results, however when tested against a fabricated design in Section 6.7 the Spectre toolchain most closely matched the empirical results. This difference is because currents below 1 pA are rarely measured and characterised in transistor models, instead being treated as negligible*. The dimensions of the starved inverters are given in Figure 6.3. The ordinary inverter’s nMOS and pMOS transistors have a width of 0.5 µm and 0.8 µm respectively and a length of 0.35 µm.

**Simulations**

To provide an indication of the behaviour of starved inverters, their performance characteristics are compared to that of a normal inverter in Table 6.2. The most

---

*Private correspondence with Dr. Paul Franzon of North Carolina State University
6.3 Starved Gates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Semi-starved</th>
<th>Fully Starved</th>
<th>Conventional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Latency (ns)</td>
<td>64,900,000</td>
<td>68,000,000</td>
<td>0.405</td>
</tr>
<tr>
<td>Rise time (ns)</td>
<td>10,100,000</td>
<td>305,000,000</td>
<td>0.660</td>
</tr>
<tr>
<td>Fall time (ns)</td>
<td>140,000,000</td>
<td>109,000,000</td>
<td>0.325</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>1.20</td>
<td>0.903</td>
<td>1.20</td>
</tr>
<tr>
<td>Minimum Output Voltage (nV)</td>
<td>9,490,000</td>
<td>8,540,000</td>
<td>10.0</td>
</tr>
<tr>
<td>Energy Per Transition (fJ)</td>
<td>163</td>
<td>52.8</td>
<td>3.15</td>
</tr>
<tr>
<td>Average Power Consumption (pW)</td>
<td>2.51</td>
<td>0.776</td>
<td>7,770,000</td>
</tr>
<tr>
<td>Leakage Power (fW)</td>
<td>15.8</td>
<td>2.14</td>
<td>37.8</td>
</tr>
</tbody>
</table>

Table 6.2: Performance characteristics of starved inverters at $V_{DD} = 1.2$ V

A striking comparison between starved and ordinary inverters is the marked difference in latency. For conventional logic where the goal is to minimise latency, gates with greater than 60 ms propagation delay are clearly undesirable. If however the goal is to create a slow waveform, their advantages become apparent.

The figures provided by dynamic energy per transition are deceptive. Although the starved inverters consume more energy, the interval over which power was integrated was vastly larger. By normalising the latency to produce the average power consumption, the energy efficiency in producing large delays becomes apparent.

The effects of supply voltage upon latency and energy per transition are shown in Figure 6.5. At the cost of an increase in power consumption, semi-starved inverters are able to attain a more stable propagation delay compared to the fully starved variants. Also of note is the greater range of supply voltages supported by semi-starved.

6.3.3 Environmental Analysis

The effect of temperature upon starved gates is plotted in Figure 6.6 and highlights an important difference between starved and ordinary gates. In a conventional
Figure 6.5: Effect of supply voltage on the behaviour of the starved inverters. Left: Propagation delay. Right: Energy per transition

Figure 6.6: Effect of temperature on the behaviour of the starved inverters. Left: Propagation delay. Right: Power consumption
inverter temperature has a minor effect, with latency increasing at hotter temperatures. In contrast, temperature has a major impact on starved inverters, with the lowest latencies occurring at the highest temperatures. Although the effect upon the energy of a single transition increases more slowly, the exponential changes in latency cause a corresponding increase in power consumption.

Figure 6.7 plots the minimum and maximum voltage levels achievable as temperature is varied. A logarithmic graph is used to show the trends, although practically after 20°C, the minimum voltage level is negligible. A chain of starved inverters will fail to propagate the signal, if the difference between the maximum and minimum achievable voltage levels is not large enough for the subsequent stage to respond to the signal. In this measure of reliability, starved gates are increasingly reliable as temperature increases.

To assess the effects of process variation on starved transitions, the circuit was simulated using the corners supplied in the process library. The following corners were tested: Typical Mean (TM=TT), Worst Power (WP=FF), Worst Speed (WS=SS), Worst One (WO=FS) and Worst Zero (WZ=SF), where SF indicates slow nMOS transistors and fast pMOS transistors.
The results of the simulation are shown in Table 6.3 for semi-starved gates and in Table 6.4 for fully starved gates.

Semi-starved gates show large differences in latency, varying from 1.02 ms to 180 ms, with differences similar in magnitude in the rise and fall time, depending on process corner. The sensitivity of latency to process variation is less pronounced for fully-starved designs, which can be attributed to the pull up network. The starved pMOS transistor is responsible for more consistent rise times, while the fall times remain as variable as the semi-starved case. The more consistent latency of the fully-starved gate is also responsible for a greatly reduced variation in average power consumption compared to the semi-starved design.

In general, fast corners have increased average power consumption, caused primarily by the reduction in latency. Fast corners reduce latency as expected, although the magnitude of the difference is significantly greater than that of a conventional inverter, and fast corners can achieve maximum or minimum voltage levels closer to the value of the supply rails.

<table>
<thead>
<tr>
<th></th>
<th>TM (TT)</th>
<th>WP (FF)</th>
<th>WS (SS)</th>
<th>WO (FS)</th>
<th>WZ (SF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Latency (ms)</td>
<td>64.9</td>
<td>1.02</td>
<td>182</td>
<td>1.04</td>
<td>128</td>
</tr>
<tr>
<td>Rise time (ms)</td>
<td>10.6</td>
<td>0.115</td>
<td>72.9</td>
<td>0.0830</td>
<td>39.8</td>
</tr>
<tr>
<td>Fall time (ms)</td>
<td>140</td>
<td>1.94</td>
<td>401</td>
<td>1.63</td>
<td>385</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
</tr>
<tr>
<td>Minimum Output Voltage (mV)</td>
<td>9.49</td>
<td>0.102</td>
<td>200</td>
<td>0.0813</td>
<td>283</td>
</tr>
<tr>
<td>Energy Per Transition (fJ)</td>
<td>163</td>
<td>213</td>
<td>102</td>
<td>192</td>
<td>96.</td>
</tr>
<tr>
<td>Average Power Consumption (pW)</td>
<td>2.51</td>
<td>210</td>
<td>0.55</td>
<td>185</td>
<td>0.753</td>
</tr>
<tr>
<td>Leakage Power (fW)</td>
<td>20.3</td>
<td>32.8</td>
<td>41.2</td>
<td>10.6</td>
<td>43.1</td>
</tr>
</tbody>
</table>

**Table 6.3:** Performance characteristics of a semi-starved inverter at various corners
### 6.4 The Buffered Starved Oscillator

<table>
<thead>
<tr>
<th></th>
<th>TM (TT)</th>
<th>WP (FF)</th>
<th>WS (SS)</th>
<th>WO (FS)</th>
<th>WZ (SF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Latency (ms)</td>
<td>68.0</td>
<td>24.5</td>
<td>152</td>
<td>29.3</td>
<td>102</td>
</tr>
<tr>
<td>Rise time (ms)</td>
<td>305</td>
<td>272</td>
<td>381</td>
<td>309</td>
<td>426</td>
</tr>
<tr>
<td>Fall time (ms)</td>
<td>109</td>
<td>7.97</td>
<td>414</td>
<td>10.8</td>
<td>443</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>0.903</td>
<td>0.989</td>
<td>0.899</td>
<td>0.896</td>
<td>1.01</td>
</tr>
<tr>
<td>Minimum Output Voltage (mV)</td>
<td>8.54</td>
<td>0.0829</td>
<td>159</td>
<td>0.0789</td>
<td>172</td>
</tr>
<tr>
<td>Energy Per Transition (fJ)</td>
<td>52.8</td>
<td>65.0</td>
<td>42.8</td>
<td>53.8</td>
<td>69.2</td>
</tr>
<tr>
<td>Average Power Consumption (pW)</td>
<td>0.776</td>
<td>2.66</td>
<td>0.282</td>
<td>1.83</td>
<td>0.676</td>
</tr>
<tr>
<td>Leakage Power (fW)</td>
<td>2.500</td>
<td>15.5</td>
<td>16.2</td>
<td>2.85</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Table 6.4: Performance characteristics of a fully starved inverter at various corners

#### 6.3.4 Tie Cells

In this section the gates of the starved transistors were tied to the voltage rails directly, however this is typically not practised in integrated circuit design. When tied directly to voltage rails, gates are vulnerable to momentary spikes in supply rail voltage, which can damage the gate oxide. To protect against this threat tie cells are used.

The disadvantage of tie cells is a slight increase in leakage power. The power consumed by a tie cell depends upon the supply voltage, but it is roughly equal to the leakage of a starved inverter. This increase can be mitigated by sharing tie cells across multiple gates and can be considered negligible if the starved logic is connected to ordinary logic at higher voltages, which will be demonstrated in Section 6.5.

### 6.4 The Buffered Starved Oscillator

Starved gates have limited value when utilised in isolation. They are slower than ordinary gates and have inferior energy efficiency. Nevertheless, there are circumstances where starved gates are beneficial. Ring oscillators, such as that depicted
in Figure 6.8 are one such example. The oscillator, referred to as a buffered starved oscillator consists of five starved gates connected in sequence.

For improved power consumption the starved oscillator contains an additional component, the buffer load. This load is placed between the output of the ring oscillator and the input of the logic controlled by the oscillator. The purpose of the load is to reduce the power consumption incurred by the oscillator. The buffer load will be described in greater detail in Section 6.5, but for now it can be abstracted as a pair of inverters.

### 6.4.1 Simulation Methodology

Simulations were performed in Cadence Spectre, using the spectre models generated for the AMS 0.35 µm process.

The testbench for the oscillator is based upon Figure 6.8, but contains an additional component connected to the output. A pair of ordinary x1 inverters, referred to as the system load, were connected in series to the output of the buffered load. The purpose of these inverters is to demonstrate that the buffered starved oscillator correctly propagates a clock signal and that the power consumption of the load is at an acceptable level. By separating the design into three components, the oscillator, the buffered load and the system load, the power consumption and output signals could be individually measured.

Because the output of the tested gates did not necessarily reach the rail voltage levels, the maximum and minimum obtainable voltage were measured. Rise time was calculated as the duration of the interval measured from the output signal
6.4 The Buffered Starved Oscillator

crossing 10% of the between minimum and maximum achievable voltage levels to
the output signal crossing 90% of this value.

To measure power consumption, the simulation window was clipped to only record
full cycles of the oscillator output, the start of which was defined by the rising cross-
ing of the midpoint between minimum and maximum achievable voltage levels. The
power consumption was then calculated by averaging instantaneous power between
these two bounds. The window calculated for the oscillator output was then used
to determine the power consumption of the buffer and system loads.

The oscillator was interpreted as successfully operating if the output of the system
load oscillated, reaching 20% and 80% of the supply voltage. Not all configurations
and environmental conditions tested result in successful oscillation. Such failures
are indicated by the lack of a point on the resulting graph. The oscillator may still
be undergoing oscillation, but if the resulting signal cannot cross the thresholds of
the subsequent loads it is of no use.

Except where otherwise indicated, all simulations were performed at a temperature
of 27°C using typical corners.

6.4.2 Simulation Results

Waveforms

After constructing an oscillator using the semi-starved gates from Figure 6.3, Fig-
ure 6.9 plots the waveform at the output of the oscillator stage and at the buffered
load stage. The slow falling transition caused by the starved pull-down transistor is
apparent, as is the fast rising transition caused by the non-starved pull-up network.
The buffered load achieves similar minimum and maximum voltage levels, but serves
to improve the slew rate of the transitions.

In contrast, consider the waveform of a starved oscillator built from the fully starved
gates of Figure 6.3 as shown in Figure 6.10. The difference between the two oscilla-
tors is that the pull-up network is now starved, and this is reflected in the output in
two ways. The slew rate for the transition is significantly slower than the previous
case, as expected. What is more noteworthy is the maximum voltage obtainable.
The starved pull-up network does not have the strength to drive the output to the
Figure 6.9: Waveform of a semi-starved oscillator

Figure 6.10: Waveform of a fully starved oscillator
rail, nor does it have the speed to reach a steady state voltage before the subsequent falling transition. The latter can be attributed to the speed at which information propagates through the oscillator. Due to the amplifying effect inherent in CMOS inverters, the next stage in the oscillator can detect the change in input before the previous stage has finished switching. What happens in this example is that the information that a switch has occurred has propagated through the oscillator and changed the input state before the gate has finished switching.

Despite this, the buffered load stage of the design is able to successfully recover a usable clock signal from the input, with sharp transitions and a satisfactory duty cycle. This restorative property allows significant flexibility in the waveform of the oscillator stage, providing the minimum and maximum voltages achievable by the oscillator are able to provide a detectable input signal.

This oscillator used for Figure 6.10 employed a starved pMOS transistor 50 µm wide. Wider pull-up transistors relative to the pull-down network allow for a larger maximum voltage and are more likely to reach a steady state voltage.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Semi-starved</th>
<th>Fully Starved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (Hz)</td>
<td>1.43</td>
<td>1.47</td>
</tr>
<tr>
<td>Rise time (ms)</td>
<td>10.5</td>
<td>233</td>
</tr>
<tr>
<td>Fall time (ms)</td>
<td>186</td>
<td>133</td>
</tr>
<tr>
<td>Duty Cycle of Load (%)</td>
<td>61.5</td>
<td>38.8</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>1.20</td>
<td>0.735</td>
</tr>
<tr>
<td>Minimum Output Voltage (mV)</td>
<td>9.5</td>
<td>9.4</td>
</tr>
<tr>
<td>Power Consumption (pW)</td>
<td>87.4</td>
<td>291</td>
</tr>
</tbody>
</table>

**Table 6.5:** Performance characteristics of starved oscillators at $V_{DD} = 1.2$ V

The performance characteristics of the semi-starved and fully-starved oscillators are reported in Table 6.5. The most striking difference is that of power consumption. The fully starved oscillator consumes over 3 times as much power as the semi-starved variant. This discrepancy will be examined in detail in later sections.
Figure 6.11: Effect of supply voltage on the behaviour of starved oscillators. Left: Frequency. Right: Combined power consumption

Environmental Behaviour

To explore the effect of supply voltage on starved oscillators, a voltage sweep was performed, with the results displayed in Figure 6.11. Both oscillators are capable of similar frequencies, although the semi-starved oscillator is capable of operating over a greater voltage range, indicated by the longer lines within the graph. Frequency varies with supply voltage, although the deviations are relatively minor. The effect of supply voltage on power consumption is much more significant. Plotted on a logarithmic axis, the power consumption of the oscillator can vary by up to a factor of one million.

The power consumption reported in Figure 6.11 represents the combined power consumption of the oscillator, the buffered load and the two inverters loading the buffer. To identify the consumer of this increase in power, the individual power consumption of the first two components were plotted separately in Figure 6.12.
low voltages, the power consumption of the oscillator and buffer load are roughly equivalent, however when voltage is increased to roughly 1 volt, the consumption of the load quickly becomes the dominant effect. The power consumption of the load will be explored in greater detail in Section 6.5.

The oscillator’s behaviour when exposed to temperature changes tells a different story. The gentle variation observed in oscillating frequency with respect to supply voltage is replaced by monotonic exponential variation. With the semi-starved oscillator experiencing greater variation compared to that of the fully-starved oscillator. If the left hand graph is inverted to obtain delay and compared to that of Figure 6.6, similar trends are observed. The presence of a crossover in the former can be attributed to the buffered load, which was not part of the Figure 6.6 test.

An important difference can be observed in the operational range. It was shown earlier that the semi-starved oscillator functioned over a larger range in supply voltages, yet when temperature is considered it is the fully-starved oscillator that functions below $10^\circ C$. 

---

---

**Figure 6.12:** Effect of power consumption on the components of starved oscillators.
Applications to Mitigate Environmental Behaviour

The significant variation of temperature in the starved oscillators can be mitigated by using the design in temperature controlled environments, for example mammalian in-vivo implants. As the core body temperature of mammals, including humans, is self-regulated, the oscillator will be protected from environmental variability. In the event that core body temperature deviates too far from the typical range, correct functionality of the implant is the least of the user’s concerns.

Alternatively, the exponential slope of the semi-starved oscillator could be a key advantage. Figure 6.14 plots the distribution of oscillation frequencies obtained as temperature was varied for supply voltages between 1 V and 1.5 V inclusive. By applying an exponential regression to the data, a formula deriving frequency from the ambient temperature was obtained, with $R^2 = 0.997$. 
6.4 The Buffered Starved Oscillator

Figure 6.14: Effect of temperature on the frequency of oscillation of the semi-starved inverter with $V_{DD}$ varied between 1 and 1.5 V.

$$T = 16.447 \ln(f) + 17.9679$$  \hspace{1cm} (6.4)

By rearranging in terms of temperature, an equation converting frequency to ambient temperature is obtained, shown in Equation (6.4). The coefficients can be determined after fabrication and programmed into a design during the testing stage. Once this is done, the system can determine the temperature at a given time simply by measuring the frequency of the oscillator against a reference clock.

Although the model works over the 1–1.5 V range studied in this chapter, greater variation is observed over a range of supply voltages, especially at low temperatures. Improving the model to account for both this and process variation would be an interesting topic for further work.

Corner Analysis

The oscillators were then subject to process variation to determine its significance using the the corners supplied in the process library. The following corners were tested: Typical Mean (TM=TT), Worst Power (WP=FF), Worst Speed (WS=SS),
Chapter 6  A Wakeup Timer Based on Starved Transistors

<table>
<thead>
<tr>
<th>Metric</th>
<th>TM (TT)</th>
<th>WP (FF)</th>
<th>WS (SS)</th>
<th>WO (FS)</th>
<th>WZ (SF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (Hz)</td>
<td>1.47</td>
<td>0.963</td>
<td>N/A</td>
<td>0.941</td>
<td>0.738</td>
</tr>
<tr>
<td>Rise time (ms)</td>
<td>10.5</td>
<td>0.115</td>
<td>N/A</td>
<td>0.00905</td>
<td>38.4</td>
</tr>
<tr>
<td>Fall time (ms)</td>
<td>15.7</td>
<td>2.22</td>
<td>N/A</td>
<td>1.85</td>
<td>4.00</td>
</tr>
<tr>
<td>Duty Cycle of Load (%)</td>
<td>63.0</td>
<td>61.7</td>
<td>N/A</td>
<td>61.0</td>
<td>69.0</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
</tr>
<tr>
<td>Minimum Output Voltage (mV)</td>
<td>9.46</td>
<td>0.168</td>
<td>1200</td>
<td>0.180</td>
<td>302</td>
</tr>
<tr>
<td>Power Consumption (pW)</td>
<td>12.8</td>
<td>577</td>
<td>0.330</td>
<td>258</td>
<td>20.2</td>
</tr>
</tbody>
</table>

Table 6.6: Performance characteristics of a semi-starved oscillator at various corners

Worst One (WO=FS) and Worst Zero (WZ=SF), where SF indicates slow nMOS transistors and fast pMOS transistors. The results of this analysis are shown in Table 6.6 for the semi-starved oscillator and Table 6.7 for the fully starved oscillator.

The first thing to note is that the oscillators tested do not support all corners at a supply voltage of 1.2 V and a temperature of 27°C. The semi-starved oscillator is unable to oscillate at the worst speed corner whereas the fully-starved oscillator cannot produce a useful clock signal at the worst zero corner. In the case of the semi-starved oscillator, the failure to operate is a consequence of the choice of supply voltage for these measurements – a usable output signal requires $V_{DD}$ to be at least 1.3 V, in comparison to value of 1.2 V used for these measurements.

As is expected, transistors with fast corners have faster transitions and are more capable of driving the output towards their respective rail. What is more surprising is the degree of variation achieved. When the worst power corner achieves a frequency 100 times greater than that of the slowest corner, the variability of the design is closer to that of that with a subthreshold supply* rather than that of a superthreshold one. This can be attributed to the reliance of both starved and subthreshold supply designs on subthreshold leakage. Of the two oscillators, the fully-starved oscillator has a more consistent performance, due primarily to the presence of a

*Section 2.2.6 on page 19
6.4 The Buffered Starved Oscillator

<table>
<thead>
<tr>
<th>Metric</th>
<th>TM (TT)</th>
<th>WP (FF)</th>
<th>WS (SS)</th>
<th>WO (FS)</th>
<th>WZ (SF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (Hz)</td>
<td>1.53</td>
<td>4.35</td>
<td>0.664</td>
<td>3.72</td>
<td>N/A</td>
</tr>
<tr>
<td>Rise time (ms)</td>
<td>212</td>
<td>94.6</td>
<td>368</td>
<td>107</td>
<td>N/A</td>
</tr>
<tr>
<td>Fall time (ms)</td>
<td>115</td>
<td>8.76</td>
<td>403</td>
<td>1.13</td>
<td>N/A</td>
</tr>
<tr>
<td>Duty Cycle of Load (%)</td>
<td>46.0</td>
<td>24.5</td>
<td>57.2</td>
<td>35.0</td>
<td>N/A</td>
</tr>
<tr>
<td>Maximum Output Voltage (V)</td>
<td>0.775</td>
<td>0.578</td>
<td>0.875</td>
<td>0.538</td>
<td>0.585</td>
</tr>
<tr>
<td>Minimum Output Voltage (mV)</td>
<td>9.46</td>
<td>0.266</td>
<td>178</td>
<td>0.152</td>
<td>559</td>
</tr>
<tr>
<td>Power Consumption (pW)</td>
<td>23.2</td>
<td>195</td>
<td>16.9</td>
<td>100</td>
<td>1.72</td>
</tr>
</tbody>
</table>

Table 6.7: Performance characteristics of a fully starved oscillator at various corners

starved pull up network. In a design based entirely on subthreshold leakage, the reduced electron mobility of pMOS transistors proves to be an advantage.

Mitigation of process variation

Due to the inability for the oscillators to function at all corners while still meeting the supply voltage requirements (1.0 V to 1.5 V), these oscillators should be taken as guidelines rather than recommendations. However, to enhance the operational range of the starved clocks, the following steps may be undertaken.

- Adjust the target supply voltage until the oscillators function. Although the majority of dies might function over a range of 1.0–1.5 V, some might be specified to reach a reduced range. Although a disadvantage, there is the possibility for product segmentation. Assuming a 1.5 V supply which decays over time, the SS semi-starved oscillator will function with a reduced lifespan. Dies in the SS corner could be marketed for applications that do not require a long operational life, for example as smart sensors for perishable foods.

- Increase the width of the pull-up or pull-down network responsible for the failure to oscillate. A common mode of failure is an inability of the oscillator to reach the voltage levels at which the load gate switches – although the output is being pulled towards a rail, the change can be propagated through
the ring oscillator and start pulling the output towards the other rail before it has reached a level at which the load switches. By increasing the width of the starved gate, the increased delay will allow additional time for the output to reach the switching threshold. Alternatively, additional stages could be inserted into the ring oscillator.

- Adjustment of either the body bias or the voltage level of the starved gate. The designs presented in this chapter assume $V_{GS}$ and $V_{GB}$ are equal to zero, however this is not a requirement. A small but positive value will increase the current flow through the starved gates, increasing the oscillation frequency and power consumption.

Comparison with published designs

The use of starved gates has previously appeared in published literature. [60] used fully starved inverters in a ring oscillator. Although based upon a similar concept, there are several differences between the design in [60] and that presented in this chapter. In addition to being implemented on a different process technology, the starved transistors in [60] had significantly longer gates with a narrow device width. In contrast, the starved transistors in this chapter have short gates but are extremely wide.

There are two primary explanations for this difference. The principle reason is the different process technology used. [60] used a 180 nm process, compared to the 0.35 µm technology used for the simulations in this chapter. One of the disadvantages of decreasing feature size is an increased susceptibility to leakage [60]. While a hindrance for most logic design, this proves to be a benefit in this instance. Starved transistors work by subthreshold leakage. With no voltage difference between the transistor’s gate and source, the only way for current to flow is through subthreshold leakage. This leakage allows the transistor to change its state and respond to changes in inputs. The greater the leakage current, the easier that this process becomes. As the rate of subthreshold conduction is poorer in the 0.35 µm technology, widening the transistors was necessary to encourage leakage while longer gates would inhibit it.

Secondly, the target supply voltage in [60] was only 0.6 V, in comparison to the design requirements of 0.8 V to 1.5 V used in this chapter. It was found that additional
width was needed for the starved oscillators to function appropriately, increasing the bounds between the maximum and minimum achievable voltage levels.

Performance comparisons between starved oscillators fabricated as a product of this research and the oscillators of [60, 62] are given in Section 6.7.1 on page 171.

### 6.5 Buffered Load

#### 6.5.1 Motivation

The disadvantages of ultra slow clocks presented in the previous section is that they can have very long transitions between the two states and that they are often unable to reach rail voltage levels. A consequence of this is that an extended period of time is spent in the short-circuit current range where both nMOS and pMOS transistors are active, resulting in elevated power consumption in the oscillator’s load.

To illustrate the problem, an ordinary minimum sized inverter was subjected to a DC voltage sweep applied to its input. The inverter was selected from a 0.35 µm logic family and simulated with a supply voltage of 0.6 V, operating in the near-threshold range for this process. The results of this analysis are shown in Figure 6.15. When the gate voltage is close to 0V or the supply voltage, power consumption is minimal, elsewhere it is several orders of magnitude higher, however still small enough to be considered negligible. The behaviour for a 1.5 V supply is similar, with a greater peak voltage, now large enough to be undesirable. If the analysis is repeated with a supply voltage of 3.3 V, the difference in power consumption becomes significantly greater and the size worst case region markedly larger. Put simply, the longer the time spent with the input between the rail voltages, the more energy is consumed.

To explain the cause of this phenomenon Figure 6.16 plots the power consumption of nMOS and pMOS transistors individually for a varying voltage applied to the transistor’s gate. When the gate voltage is at either extreme, only one transistor is active. When combined in a CMOS gate, for example an inverter consisting of an nMOS and a pMOS transistor, the large current passed by one transistor does not pose a problem as the other one limits the current flow and is approximated as an open circuit. However should the gate voltage be in the middle of the range both
Figure 6.15: Power consumption of an ordinary inverter with a DC voltage sweep applied to the input.

Figure 6.16: Power consumption of individual transistors after application of voltage to gate
transistors are active. In this situation, a connection is created between the supply and ground. This allows a large amount of current, known as short circuit current, to flow. In a conventional CMOS gate this state can occur briefly while the gate is switching from one logic state to another, with the fast switching speed limiting the energy consumed. For gates with very slow transitions, this can be a significant problem, with the power consumption of the load gate vastly greater than that of the slow input gate. If the input gate cannot achieve rail-to-rail swing, the problem is even more serious as a higher proportion of time is spent in this high current region.

6.5.2 Solutions

In a CMOS design, worst case power consumption is attained when both pull up and pull down networks are activated. In this situation, the transistors in the critical current path between the supply and ground rails are operating in the saturation region. It is within this region that insight shall be obtained. [83] provides the model for the drain current of a MOSFET operating in the saturation region, shown in Equation (6.5).

\[ I_D \approx \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{GS} - V_{TH})^2 (1 + \lambda V_{DS}) \]  

(6.5)

To reduce power consumption, it is sufficient to reduce the drain current. This can be accomplished in two principle ways.

Increasing Gate Length

The length of a transistor’s gate, \( L \) influences the drain current in two principle ways. Primarily drain current is inversely proportional to length. Its secondary effect is caused by channel length modulation, denoted as \( \lambda \). Increasing the gate’s length decreases channel length modulation, causing a further reduction in drain current.

The long inverter to be tested has an nMOS length of 20 \( \mu \)m and a pMOS length of 10 \( \mu \)m. In the same way that different width ratios can change the crossover point at which a gate switches from high to low voltage, variations in the length ratios can
have the same effect. This particular ratio was chosen after trial and error because of its suitability for semi-starved oscillators, however other ratios are equally valid.

**Increasing the threshold voltage, \( V_{TH} \)**

The second method to reduce drain current involves minimising the term \((V_{GS} - V_{TH})^2\). As the transistor is operating in the saturation region, \(V_{GS} > V_{TH}\), this can be accomplished by reducing \(V_{GS}\) or increasing \(V_{TH}\). However only the latter is practical. Although we are considering the equation for a single transistor, the equation needs to be applied to transistors in both the pull up and pull down network. If the input were to be modified to reduce \(V_{GS}\) for one of the networks, it will result in an increase in the \(V_{GS}\) of the transistors belonging to the other network. As a result, improvements are limited to an increase in the threshold voltage.

An additional benefit to high threshold voltage transistors is reduced leakage, although this comes at the cost of reduced performance in the form of longer propagation delays. This issue is a disadvantage for general purpose logic, but in the context of a load device for an ultra-slow clock it is a minor concern. The transition times and periods of that clock are sufficiently slow as to make the increase in propagation delay negligible in comparison.

A more serious problem is the change in the transition voltages between the cutoff, triode and saturation regions. For the transistor to be active (triode or saturation region) it requires \(V_{GS} > V_{TH}\), providing an upper bound on the threshold voltage. Therefore it will be necessary to perform the simulations with the starved oscillator input to ensure that correct functionality is maintained.

Unfortunately high threshold voltage transistors are not always available within process libraries. For example, the AMS 0.35 \(\mu\)m process used to generate the results for this chapter does not support dedicated high threshold transistors. It does however support high voltage transistors, which are capable of operating at 5 V supply voltage rather than the standard 3.3 V supply, so these transistors were used as a proxy.
6.5 Buffered Load

![Testbench for load gate voltage sweep](image)

6.5.3 Comparison

Ideal Input

The first step is to test the behaviour of the buffered load against an ideal input. In the testbench depicted in Figure 6.17, a DC input was applied to a chain of four inverters. The first half of the chain consists of the buffered load, which was tested in different configurations. The second half consisted of two ordinary x1 inverters in sequence to ensure that the buffered load succeeded in reducing the power consumption for the subsequent stage. A DC voltage sweep was then applied to the input, ranging from 0 V to the supply voltage. The performance of the buffered load was evaluated by measuring the power consumption of all four inverters as the input voltage was varied. A baseline buffered load was tested to allow the effects of each buffered load to be determined. The inverters in the baseline load had the same width as the other buffers tested (1 µm), but used standard threshold transistors and minimum length (0.35 µm).

The resulting voltage sweep is shown in Figure 6.18. The graph shows that as load devices, Schmitt triggers do not improve power consumption. The sharp drop in the Schmitt trigger’s power consumption can be attributed to the hysteresis property of the trigger. Although long gates are capable of a significant reduction in power consumption compared to baseline, high threshold gates are even better at low power operation. A combination of the two gives even better results. The highest performing load has a peak power consumption approximately 100,000 times lower than that of the lowest performing gate, highlighting the need for the buffered load.

In Figure 6.19, other lengths are simulated for the buffered load. There is a significant difference between the ordinary inverter and the gates with length of 5 µm, although the rate of change decreases with larger lengths. This is anticipated by
Figure 6.18: Power consumption of various load gates with a DC voltage sweep applied to the input at $V_{DD} = 1.2$ V

Figure 6.19: Power consumption of inverters with various lengths with respect to gate voltage.
Equation (6.5), where \( I_D \propto \frac{1}{L} \). In the case of an unequal ratio between the length of the pull-up and pull-down networks, power consumption behaves as a composite of gates with those lengths considered separately.

**Realistic Input**

The next step is to evaluate the performance of the buffers with a realistic input, that of the oscillators presented earlier. This is important for two reasons. Firstly, the oscillator output is not guaranteed to reach the supply and ground rails, leading to elevated power consumption. Secondly, the power consumption of the load cannot be independently assessed without determining the proportion of time spent in high current flow regions. To determine the relative merits of the buffered loads, the testbench was simulated within Cadence Spectre undergoing a supply voltage sweep at 27°C. The power consumption of all components was measured in aggregate, and is reported in Figure 6.20 for semi-starved gates and Figure 6.21 for fully starved gates.

Unlike the ideal gate voltage sweep performed earlier, these measurements were performed with multiple supply voltages. Doing so reveals that the superior buffered
load in terms of power consumption depends on the supply voltage. Although Figure 6.18 showed the long high threshold buffer had the lowest peak power consumption, in a realistic situation it is only beneficial at high voltages.

### 6.5.4 Disadvantages of Long Gates

Long gates come at a cost. Compared to an ordinary inverter with minimum length transistors ($1 \lambda$), an inverter with transistors $29 \lambda$ long has a latency which is 251 times larger with a 1.5 V supply. This, along with a lack of improvement in energy efficiency make long inverters impractical for general usage. However, when used as load devices for starved inverters, these disadvantages are not a problem. Starved inverters already have exceptionally long latencies (1.4 million times that of the long inverter), so the additional latency cost of a long inverter is negligible in comparison. Even if this was not the case, the additional delay would only result in a change of phase in the clock signal which will not be detectable if the reference clock is defined as the output of the buffered load.
6.6 Starved Oscillator Design Space Exploration

6.5.5 Recommended Buffer Loads

Both the long buffer and the high threshold buffers are suitable for use in a design. The recommended buffer load for a given situation depends upon the nature of the oscillator input, the supply voltage at which it is to be used and the availability of high threshold gates in the target process. The remaining sections of this chapter will continue to use the long inverter buffer as the focus is shifted to the oscillator stage.

One point that has not yet been considered in this section is the influence of the buffer’s threshold voltages and process variation. The output of starved oscillators is not guaranteed to reach rail levels, especially when the process corner tends away from typical mean. To ensure successful operation, the buffer load needs to be tolerant to variations in process variation and in the achievable maximum and minimum voltage levels of the input oscillator stage. The choice of a buffered load might then be determined by the ability to generate appropriate output over the desired range of operating conditions, rather than a selection on power consumption alone.

6.6 Starved Oscillator Design Space Exploration

Section 6.4 presented two ways to design a starved oscillator, but there are many ways in which it can be done. This section seeks to explore the various possibilities.

6.6.1 Sizing

To explore the effects of sizing of the starved transistor, the two oscillators were simulated with several different widths. The x-axis of the graphs in this section depict the starved nMOS transistor’s width. In the semi-starved case, the always-on pull-up transistor had the same width, whereas with the fully-starved oscillator the width of the pMOS starved transistor was increased to 150% of the width of the nMOS starved transistor.

Figure 6.22 shows the effect of width of the transistors on frequency. In all cases, larger widths resulted in a slower rate of oscillation. This can be attributed to the additional capacitance present in each stage of the oscillator caused by the width.
increase. At 1.5 V, the fully-starved oscillator is unable to generate a useful signal when the nMOS starved transistor width is 30 µm, yet is able to do so when the width is increased.

It is when considering power consumption that sizing becomes more interesting. When the supply voltage is equal to 1 V, the power consumption of the semi-starved oscillator increases and that of the fully-starved oscillator decreases, as shown in Figure 6.23. If $V_{DD}$ is set to 1.5 V, the trend is reversed.

In the case of the semi-starved oscillator, at both supply voltages the power consumption of the ring oscillator increases linearly as widths increase and the power consumed by the load decreases. At a supply voltage of 1.0 V, the power consumption caused by the ring oscillator and the buffered load are approximately equal, but the rate of change in the ring oscillator’s power consumption is greater than that of the buffered load, so power consumption increases. At $V_{DD} = 1.5$ V, the proportion of total power consumption caused by the load becomes dominant, hence the decrease in total power usage.
6.6 Starved Oscillator Design Space Exploration

![Graph showing the effect of starved transistor width on power consumption.](image)

**Figure 6.23:** Effect of starved transistor width on power consumption. Left: $V_{DD} = 1.0 \text{ V}$. Right: $V_{DD} = 1.5 \text{ V}$

### 6.6.2 Balancing pMOS transistor

The semi-starved design presented in Section 6.3 included a permanently active pMOS transistor, which will be referred to as a balancing pMOS transistor. The purpose of this device is to improve the behaviour of the oscillator. In Figure 6.24, a semi-starved oscillator was simulated without this transistor. This configuration will be referred to as *unbalanced*.

Whereas in the balanced case the oscillation frequency is approximately flat over an interval of 2 V, the unbalanced case experiences increasingly faster frequencies. Also of note is the smaller upper bound of supply voltage which allows the clock signal to propagate to the load gates. As a further test, an unbalanced oscillator with starved nMOS width of 150 µm, was simulated to see if larger values could compensate for the lack of a balancing transistor. At low voltages, the frequency of operation of the wide unbalanced configuration is similar to that of the balanced case, but beyond this the rapid increase in frequency still occurs. A small reduction in power consumption in the configuration using the balancing transistor is also observed, but
Figure 6.24: Frequency and power consumption of semi-starved oscillators tested with and without balancing pMOS transistor.

this can probably be attributed to the reduction in frequency of oscillation rather than any inherent improvements in efficiency.

6.6.3 Starved Placement

The possible configurations of semi-starved oscillators allow flexibility in the placement of starved and balancing transistors. The devices can either be placed at the rail, for example between $V_{DD}$ and the pull-up network, or at the output, which is the configuration previously examined. In the latter case, to have any effect on the circuit the balancing transistor needs to be tied to the input voltage of the gate, rather than the appropriate rail in the former configuration. The frequency and power consumption of three such variants are shown in Figure 6.25.
When the pMOS balancing transistor is fixed at the output, the difference between rail-starved and output-starved nMOS transistors is not significant. In comparison, when the balancing transistor is placed at the rail, the oscillation frequency is markedly faster, but can only propagate a clock signal over a narrower voltage range.

The large increase in frequency in the rail balanced shows similarities to an unbalanced oscillator which is otherwise equivalent. The two cases are plotted in Figure 6.26. Although there is a significant difference in the range of voltages at which the clock signal can propagate, the two oscillators are closely matched with respect to frequency. This suggests that the balancing transistor is only effective when placed at the output.
This is not surprising, given the balancing transistor is always operating in the saturation region with an input that doesn’t change. Once the transistor reaches steady state, the internal capacitances behave as open circuits, neither charging or discharging. The transistor can then be modelled as a short circuit, approximating the unbalanced design. The small differences between the rail balanced gate and the unbalanced design are likely due to higher order effects, not accounted for in the short circuit model. Should the gate of the balancing transistor be connected to the output of the previous stage, rather than to the ground rail, it will have a more significant effect on the circuit’s behaviour.

[60] reported a reduction of 19.6 % in rise and fall time through the use of output starving compared with rail starving for their 180 nm fully-starved design. In the configuration with nMOS starved at the rail and pMOS balancing transistor at the output, an average improvement in fall time of 3.5 % was recorded, varying with supply voltage from -10% to 10% when compared to an oscillator with both transistors at output. When the latter configuration was tested against an oscillator with starved and balancing transistors placed at the rails, the output starved variant
exhibited on average 112% slower rise times and 94% slower fall times. The fully starved oscillator was unable to oscillate successfully after conversion to rail starving.

6.6.4 Stack Depth

The starved gates from Section 6.3 included two transistors connected in series in both the pull up and pull down networks, the gates of which were connected to the input. This was selected for the superior power efficiency observed in simulations compared to variants with stack depths of 1 or 3 transistors. At 1.5 V, a semi-starved oscillator constructed of two stack inverters consumed 77% and 72% of the power consumption of oscillators constructed of one stack and three stack inverters respectively. A potential disadvantage is an increase in oscillation frequency of up to 10%.

6.6.5 Additional Resistive Load

The 50 μm semi-starved oscillator presented earlier is not able to operate above approximately 2.75 V. To explain the reason for this, Figure 6.27 plots the minimum
voltage achievable by the oscillator stage. When the supply voltage is increased beyond 1.75 V, the minimum achievable voltage also increases. Above approximately 3.2 V, the minimum voltage is too high to properly trigger the transition of the load stage, resulting in a failure for the oscillation signal to be propagated.

The inability to oscillate at high voltages is not the only disadvantage. As the minimum voltage increases, the period of time when both pull-up and pull-down networks of the load gates are active also increases, leading to increasingly larger power consumption at higher supply voltages.

To reduce power consumption and increase the range of operational frequencies, it is necessary therefore to encourage the output of the oscillator stage to settle on a lower minimum output voltage. One way of doing this is to introduce a pull-down resistor to the output of the oscillator stage, as shown in Figure 6.28. The pull-down resistor is a commonly used technique in electronic design to drive an output to a safe value when the output is not being driven. To minimise power consumption, it is necessary for the load resistor to be as large as possible to minimise the current flow between supply and ground when the pull-up network is active.

Using a 10 GΩ resistor, the oscillator with resistive load was simulated. The results are shown in Figure 6.29, compared against an oscillator without the resistive load. There are three aspects of note. Firstly, the oscillator with resistive load can support higher supply voltages, although the minimum supply voltage required to function also increases. The failure to function at low voltages can be attributed to the inability of the pull-up network of the oscillator to contend with the resistive load driving the output down.
Secondly, there is an increase in frequency of approximately 0.4 Hz which is common over the entire range. The pull-down resistor is more effective at driving the output down than the starved pull-down network, resulting in a faster transition for the ring oscillator stage subsequent to the output. Finally there is a significant reduction in power consumption, up to 94% lower at $V_{DD} = 2.5$ V.

Despite the promise of the resistive load in semi-starved oscillators, attempts to introduce it into fully-starved variants were unsuccessful. Tests were performed with both pull-up and pull-down load resistors, but although they succeed in driving the output to the appropriate rail, no oscillation was observed.

In order for the resistive load to be effective, it needs to be extremely large to avoid increasing power consumption through the connection between the active pull-up network and ground. This large size makes it impractical to include a resistor in a
CMOS design. In a 0.35 µm process, a resistor made of high resistance polysilicon would be somewhere in the vicinity of 5 m long∗ if stretched out linearly.

### 6.7 Fabrication

Two oscillators, both consisting of 5 stages were selected for fabrication. The configuration of the oscillators are shown in Figure 6.30. The load resistor was connected to the oscillator using an analogue pad. Note that in Oscillator B the balancing pMOS transistor was omitted from the NAND2 gate. This had the effect of increasing the performance difference between the two oscillators.

The oscillators were fabricated in a AMS 0.35 µm process with four metal layers, using design files supplied by CMP. A micrograph of the fabricated oscillators is shown in Figure 6.31. Omitted from the micrograph are the analogue pads used for

---

∗The confidentiality of process parameters prevents a more precise answer.
6.7 Fabrication

Figure 6.31: Micrograph of fabricated oscillators (from Kitchener and Phillips [51])

connecting the resistive load. To provide an indication of scale, the starved inverter used in Oscillator A, had an area of 621 \( \mu \text{m}^2 \), in comparison to that of 44.4 \( \mu \text{m}^2 \) for an x1 sized inverter. To allow for the measurement of power consumption, each oscillator, including its buffered load, was allocated a separate power rail, which was connected to an analogue pad. I have previously published the oscillator designs, along with their results, in [51].

6.7.1 Experimental Results

Methodology

Power consumption of the oscillator and buffered load was captured using a Keithley 236 Source Measure Unit connected to oscillator’s power rail. A secondary power supply set to the same voltage level was used to power the remainder of the chip. A LeCroy Waverunner 6100A oscilloscope was used to measure the output signals.
Due to the large effect that temperature has on oscillator frequency, particular care was taken for the empirical measurements. A thermometer was used to measure the ambient temperature for each data point, to ensure that results were only compared to those with a similar temperature. Each set of results was recorded during one sitting to minimise the effects caused by a change in temperature. The comparison simulations were performed at the same temperature as the experimental results. In the accompanying simulations no attempt was made to match the process corner to that of the fabricated designs. The significant effect of process corner on oscillation frequency\(^\ast\) makes it unlikely that the two sets of results will be closely matched, but they are sufficiently alike so that the trends can be observed.

**Testing Complications**

Without the resistive load connected, it was observed that the oscillators suffered significantly from noise, particularly from a mains power induced 50 Hz oscillation. The cause of this was that the bond-wire and package were acting as an antenna and severing the bond-wires solved the problem. Although this removes the noise source, the additional capacitive loading caused by the analogue pad is still present, which will influence the results. The action of severing the bond-wires is permanent, as reconnecting them is impractical.

**Results**

Figure 6.32 displays the effect of supply voltage on oscillation frequency. Results from two other oscillator designs have also been included, the Lin Oscillator [62] based on gate leakage and the Lee Oscillator [60], which is also a starved design.

The performance of the fabricated oscillators with pad load closely match their simulated counterparts. The presence of the analogue pad significantly reduces the maximum voltage at which a clock signal can propagate to the output, but the trends are similar. The increased frequency of the resistive load cases behaves as expected due to the faster action of the pull-down network.

A greater difference between the output starved Oscillator A and the rail starved Oscillator B is observed, compared to the simulations performed in Section 6.6.3,

\(^\ast\)See Section 6.4.2
Figure 6.32: Supply voltage vs frequency of fabricated oscillators (from Kitchener and Phillips [51])

however this can be attributed to the lack of a balancing transistor in the NAND2 gate.

The power consumption of the oscillators is plotted in Figure 6.33. The significant effect of supply voltage on power consumption matches what was expected, although the greater difference between simulation and pad load results can be attributed to the large capacitive effect of the analogue pad. The benefit of the resistive load at high supply voltages matches the results from Section 6.6.5, although the inferiority at lower supply voltages is unexpected but not unreasonable.

Also of note is the rate of increase in power consumption is not as steep as that of the Lin Oscillator and that the proposed oscillators can achieve an equivalent level of power consumption at a significantly greater supply voltage.
Chapter 6  A Wakeup Timer Based on Starved Transistors

Figure 6.33: Power consumption vs frequency of fabricated oscillators (from Kitchener and Phillips [51])

6.8 Application to Subthreshold Logic

From a high level perspective the starved oscillators and the subthreshold logic studied earlier in this thesis operate on similar principles. Both technologies rely on subthreshold leakage and both will show a decrease in latency or period if steps to induce leakage are undertaken. Despite this, the two technologies cannot easily be incorporated as the starved oscillators require a higher supply voltage to operate. Although the fabricated starved oscillators began to oscillate at approximately the same supply voltage at which the FIFOs minimised their energy per operation*, there may not be enough of a margin to allow for acceptable yields.

The prospect of integration changes if multiple supply voltages are introduced - the wakeup timer could be controlled by the higher voltage compared to the subthreshold

*Section 5.5.5 on page 116
or near-threshold logic. Both supply voltages do not need to be used at the same time – as the purpose of the wakeup timer is to operate whilst the remainder of the circuit sleeps, the subthreshold section could be powered off when the supply voltage is elevated. The challenge would then be to design power management logic capable of switching between the two states, while still having a negligible level of power consumption.

Assuming that both subthreshold logic and the starved oscillators could function simultaneously, the latter would not be recommended for use as the clock for synchronous regions, as the lack of post-fabrication configuration makes adjustments to account for process variation impractical. Because variation at subthreshold does not have a significant spatial correlation\[26\], one cannot rely on a clock being afflicted with similar variation to the logic that it controls. Instead, the clock would need to have an adjustable frequency which can either be self-calibrating or determined during post fabrication tests. The design of such a variation aware clock would be an interesting topic for further research.

### 6.9 Further Work

The starved gates examined in this chapter were implemented on an old, mature process. Before starved transistor oscillators are implemented in more modern technologies, their suitability to newer processes needs to be examined. As leakage increases, starved transistors switch more quickly and are more robust to environmental variation. An untested hypothesis is that smaller processes with increased subthreshold leakage compared to the tested 0.35 \( \mu \text{m} \) process would result in more reliable starved transistors that do not need extreme widths. In evidence of this hypothesis, [60] implemented starved transistors in a 180 nm process. Compared to the implementation proposed in this chapter, theirs achieved superior swing using significantly narrower transistors.

The fabricated oscillator designs provided promising results which warrant a more rigorous assessment of the effects of process variability on the fabricated oscillators. This would require the fabrication of a large number of chips to produce a statistically meaningful population. Fabrication is required as the simulation tools have not been strongly validated for the currents produced by the oscillators.
Even when the bond-wires were disconnected, the pad contributed additional capacitance, significantly increasing the cost of switching. Were a new chip to be fabricated to test starved designs, the analogue pad would be omitted.

6.10 Conclusion

This chapter provides an introduction to starved oscillators, examining their possible configurations and performance over a wide range of operating conditions. It is not possible however to provide recommendations as to the optimum design. The best oscillator for a particular application depends on the required specifications and the environment in which it will operate.

6.11 Acknowledgement

The experimental results obtained from the fabricated oscillators have previously been published in Kitchener and Phillips, *Starved picowatt oscillator for remote sensor wake-up timer*, Electronics Letters, © 2012 IET [51].
Chapter 7

Conclusions

This thesis examines the properties of logic operating with subthreshold and near-threshold supply voltages, with particular emphasis on the impact of process and environmental variation.

The primary motivation for the use of subthreshold or near-threshold supply voltages is a reduction in power consumption. Chapter 2 demonstrates this at the gate level, but the power savings come at the cost of a significant increase in latency\(^*\). Operating at subthreshold supply voltages is further disadvantaged by a marked susceptibility to process variation, which affects both the latency of gates\(^†\) and the minimum supply voltage required for successful operation\(^‡\).

The effects of subthreshold operation on the performance of larger systems is examined in Chapter 3, with a case study on the design of adders. More complicated adder designs can achieve lower latency at the cost of increased area overhead and reduced power efficiency\(^§\). Section 3.3.4 on page 46 poses the question of whether a fast, but energy inefficient, adder architecture can provide a reduction in total system power consumption by reducing the length of the critical path. Although a slight reduction in overall power consumption is observed, the effects are negligible for all but the largest designs. Chapter 3 concludes with the observation that in the subthreshold region, simpler designs are better providing timing specifications can still be met.

Chapter 4 address architecture design from a different perspective, that of reliability. Large, but uniform, changes in process variation can be tolerated by a reduction in the speed at which a system operates. Unfortunately, process variability is not

\(^*\)Section 2.2.2 on page 11
\(^†\)Section 2.2.6 on page 19
\(^‡\)Section 3.2.4 on page 39
\(^§\)Section 3.3.4 on page 46
uniform, and intra-die variation has a severe effect on subthreshold systems. Consequences of this include violations of setup and hold time constraints for flops and changes in the critical path, affecting both synchronous and asynchronous systems. To mitigate intra-die process variation, asynchronous design methodologies are proposed. Although it is not possible to formally prove timing correctness, quasi-delay insensitive circuits can be designed to tolerate process variation to a sufficient level of probability\(^*\).

Designing entirely asynchronous systems may not be practical. The development tools are not as capable or refined as those for synchronous methodologies and the ability to incorporate third-party IP is restricted. To address this, Chapter 4 proposes that the Globally Asynchronous Locally Synchronous approach should be applied at Subthreshold, allowing the incorporation of synchronous elements into an otherwise asynchronous design. To help achieve this, example interfaces for use within a GALS system are presented and evaluated\(^\dagger\).

Chapter 6 addresses power efficiency from a new perspective. The power consumption of any device can be reduced by powering off the system when not in use. In order to continue to perform useful work when needed, a timer is necessary to wake the system. To address this need, two new types of gates are introduced – starved and long-length. In ordinary usage they are inferior to conventional designs with greatly increased latency and reduced power efficiency, but when incorporated into a wakeup timer, the high latency proves advantageous, achieving long periods and low power consumption\(^\ddagger\).

Despite the ability to operate at higher supply voltages, wake-up timers built from starved oscillators share the disadvantage of heightened susceptibility to process variation with subthreshold logic, affecting both the frequency of oscillation and the range of supported operating voltages\(^\S\). With the validation of the simulated results against fabricated oscillators\(^\z\), starved wake-up circuits are a viable way to achieve energy efficiency and prolonged battery life.

\(^*\)Section 5.3.3 on page 105  
\(^\dagger\)Chapter 5  
\(^\ddagger\)Section 6.3.2 on page 135  
\(^\S\)Section 6.3.3 on page 136  
\(^\z\)Section 6.7 on page 169
As the demand for thinner electronic devices limiting the size of batteries and the slow improvements of battery technology, increases in the energy efficiency of electronic circuits are critical. Reducing the supply voltage to subthreshold or near-threshold levels or incorporating starved transistors into slow oscillators are two ways to achieve this goal. With further research into methods to mitigate the effects of process variability, these techniques could provide breakthroughs in power efficiency and allow the development of new applications for electronics.
Bibliography


BIBLIOGRAPHY


